SAND REPORT

SAND2004-0101
Unlimited Release
Printed January 2004

Simulating Economic Effects of Disruptions in the Telecommunications Infrastructure

Dianne C. Barton, Eric D. Edison, David A. Schoenwald, Roger G. Cox, and Rhonda K. Reinert

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865)576-8401
Facsimile: (865)576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.doe.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800)553-6847
Facsimile: (703)605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2004-0101
Unlimited Release
Printed January 2004

Simulating Economic Effects of Disruptions in the Telecommunications Infrastructure

Dianne C. Barton, Eric D. Eidson, David A. Schoenwald,
Roger G. Cox, and Rhonda K. Reinert

Evolutionary Computing and Agent-Based Modeling Department
Sandia National Laboratories
Box 5800
Albuquerque, NM 87185-0318

Abstract

CommAspen is a new agent-based model for simulating the interdependent effects of market decisions and disruptions in the telecommunications infrastructure on other critical infrastructures in the U.S. economy such as banking and finance, and electric power. CommAspen extends and modifies the capabilities of Aspen-EE, an agent-based model previously developed by Sandia National Laboratories to analyze the interdependencies between the electric power system and other critical infrastructures. CommAspen has been tested on a series of scenarios in which the communications network has been disrupted, due to congestion and outages. Analysis of the scenario results indicates that communications networks simulated by the model behave as their counterparts do in the real world. Results also show that the model could be used to analyze the economic impact of communications congestion and outages.

Acknowledgments

We would like to extend our appreciation to Dr. Richard Pryor, the creator of Aspen, who has worked with us in developing CommAspen. We also thank Mark Ehlen for providing expertise in the economics of the banking system.

Contents

Figures

Tables

Nomenclature

ACK-M	message acknowledgment
ACK-P	packet acknowledgment
Aspen	(an agent-based simulation model of the U.S. economy)
Aspen-EE	Aspen Electricity Enhancement
GALCS	genetic algorithm learning classifier system
LDRD	Laboratory Directed Research and Development (a Sandia program)
MVD	Motor Vehicle Division
NISAC	National Infrastructure Simulation and Analysis Center
N-ABLE	NISAC Agent-Based Laboratory for Economics
Sandia	Sandia National Laboratories
SET	severely errored time step
TCP	Transmission Control Protocol

CommAspen: Modeling Infrastructure Interdependency through Communications

Introduction

Individually and collectively, we depend on critical infrastructures in the United States to provide the essential services that support (among other things) our economic prosperity, quality of life, and national security [1]. These infrastructures have recently been categorized into the following sectors: agriculture, food, water, public health, emergency services, government, defense industrial base, information and telecommunications, energy, transportation, banking and finance, chemical industry and hazardous materials, and postal and shipping [2]. While historically these infrastructures have been vulnerable to malevolent acts and to natural disasters, today, these systems are more threatened by the increasing complexity inherent in their growing interconnectedness and interdependence [3]. The telecommunications infrastructure is the link that allows interconnection between all other infrastructure networks and enables the transfer of goods, services, and information. Communication literally constitutes both the foundation and an upper limit to the economic functioning of a society.

Increasing interconnectivity is being driven by modern business trends that rely heavily upon telecommunications for management and operation. On balance, interconnectivity will improve our nation's economic efficiency; however, tight coupling between infrastructures also results in situations where a disturbance in a formerly isolated infrastructure unexpectedly cascades across diverse and seemingly unrelated infrastructures. In simulations using Sandia National Laboratories' (Sandia's) Aspen Electricity Enhancement model, i.e. Aspen-EE, this effect is observed when policy decisions lead to power disruptions that cause second-order impacts on pricing trends. The interconnection of these infrastructures creates tremendous interdependency and vulnerability.

Because interdependent infrastructure networks are complex systems, they do not readily submit to traditional reductionist analysis, where an individual infrastructure would be examined in isolation from other infrastructure elements. The reductionist approach ignores that linkages between infrastructure elements do exist and that such linkages cannot be deduced from isolated analysis. Complex systems exhibit a rich variety of behaviors, including many that are counterintuitive. To capture these effects, we must view complex systems from a holistic rather than a reductionist point of view.

Agent-Based Approaches for Complex Systems

To analyze interdependent infrastructure systems in a more holistic way, Sandia and other research institutions have developed models of critical infrastructure systems using agent-based approaches. Sandia's first agent-based model of the U.S. economy,

developed in the mid-1990s, is called Aspen. This model is a Monte Carlo simulation that uses agents to represent various decision-making segments in the economy, such as banks, households, industries, and the Federal Reserve. An agent is a computational entity that receives information and act on its environment in an autonomous way; that is, an agent's behavior depends at least partially on its own experience. Through the use of evolutionary learning techniques, Aspen allows us to examine the interactive behavior of these agents as they make real-life decisions in an environment where agents communicate with each other and adapt their behaviors to changing economic conditions, all the while learning from their past experience. In 2000, Sandia developed a new model of infrastructure interdependency called Aspen-EE. This model extended the capabilities of Aspen to include the impact of market structures and power outages in the electric power system, a critical infrastructure, on other infrastructures in the economy [4].

One of the limitations of agent-based models in current development at Sandia and other research institutions is that communication is treated simply as message passing between agents. Effectively, the telecommunications infrastructure is not specifically represented. None of the models simulates the differences in communication over telephone, computer, wireless, or other networks and therefore cannot model the impact of specific communication failures on the whole system. Nor can current models simulate the impact of other infrastructure failures on telecommunications.

To address the communications deficiencies described above, Sandia revised and restructured the Aspen-EE model to include a more realistic representation of the telecommunications infrastructure. This new model of infrastructure interdependency is called CommAspen. In CommAspen, communication is treated as an integrated agent system capable of creating, transforming, sending, receiving, and storing information and messages over time and across distance. With CommAspen, we can model communication networks or medium-specific vulnerabilities to failures and their dependence on supporting infrastructures like power.

Overview of This Report

This report describes the capabilities and uses of CommAspen in supporting the analysis needs of infrastructure protection specialists. In this first section, we have provided the background that led to the development of the model. Next, we present a brief discussion of the preliminary research conducted to characterize the telecommunications industry as a critical component of the model. The third section of the report gives a detailed overview of the new model, including its architecture, mechanics, input and output files, agent composition and behavior, and communications network. The fourth section addresses characteristics of the problem we are using to demonstrate the capabilities of CommAspen's communications network. The interdependence between telecommunications and banking is highlighted in this problem. Results and analysis of a set of sample scenarios for the problem are presented in the fifth section of the report. Finally, we discuss other applications of CommAspen and identify future enhancements for the model. For those interested in understanding the input

details, Appendix A describes the layout and contents of the input file to CommAspen, and Appendix B lists a sample input file.

Approach to Developing Communications Agents

In the early stages of developing CommAspen, we sought to gain a greater understanding of the key aspects of the telecommunications industry. We looked at the structure of the industry, how it is regulated, who its major equipment suppliers are, what pricing models it employs, what architectures are in use or anticipated, and how telecommunications is interdependent with other infrastructures. Based on this research, we then proposed a set of categories of telecommunications agents for the model and identified characteristics of the services these types of agents could provide. As disruption of telecommunications services is of prime importance in developing a realistic model of infrastructure interdependency, we also postulated the kinds of behaviors that might be expected during and after a telecommunications outage by both consumers of telecommunications services and providers of these services.

Our work on defining the behavioral characteristics of telecommunications agents for CommAspen was interrupted by the events of September 11, 2001. However, through press accounts post-9/11, we were able to collect evidence about actual responses to telecommunications outages by both consumers and providers of telecommunications services. We compared these actual responses to those predicted pre-9/11, as a means of validating our initial preconceptions of agent behaviors.

The research and analysis discussed above are described in the report entitled *A Year 2003 Conceptual Model for the U.S. Telecommunications Infrastructure* [5]. For this initial version of CommAspen, we implemented a few of the agent behaviors, but by no means all. The report serves as a guide for the further development of the telecommunications infrastructure in CommAspen.

The Model

The development of CommAspen was funded by Sandia's Laboratory Directed Research and Development (LDRD) program. In building CommAspen, we used the agent-based model Aspen-EE as the foundation and completely revamped its architecture. CommAspen was written in C++ to run on a single-processor-machine. A version of CommAspen for parallel-processor machines is planned.

Agent-based models assume that complex behavior emerges from many individual, relatively simple interactions rather than from the complexity inherent in any particular agent. Agents have simple rules of behavior and react to their environment (i.e., the other agents and any static features) without reference to any global goals—in other words, the agents are undertaking purely local transactions. The net results of these local interactions

and decisions are phenomena that emerge on a global level. When unexpected results emerge from the simulation, it is important to be confident that we understand the fundamental processes built into the model. The complexity of agent-based modeling should be in the results of the model and not in its assumptions.

Our detailed description of CommAspen begins with a brief overview of how Aspen-EE is structured to lay the groundwork for the changes in CommAspen. Next, we explain the new architecture in CommAspen as it relates to agents and their interactions with their environment. A brief summary of differences between Aspen-EE and CommAspen is included. The discussion then addresses the mechanics of the model and identifies the input and output requirements. Following is a close look at the components of user-constructed agents and the rules of behavior that are both common across, and unique to, different types of agents. Interrelationships both internal and external to agent types are then identified, capturing aspects of the complexity in the new architecture. The next topic introduces the features of the bulletin board, a type of agent that provides pricing information to the other agents during a simulation. The section concludes with a discussion of how the communications network developed specifically for CommAspen works.

Aspen-EE: The Foundation

Revising Aspen-EE to become CommAspen entailed a major restructuring effort. In Aspen-EE, the major model components are a set of agent types (or classes), as shown in the inset to the right. For example, in Aspen-EE there is a class of agents for households, a class of agents for industry, a class of agents for government, etc. In total, Aspen-EE has 10 classes of agents. The behavior of individual agents in a particular class is defined by a set of decision rules. Thus, for example, in Aspen-EE all individual households in a simulation will behave the same according to the decision rules for the household class; all individual industry agents will behave the same according to the decision rules for the industry class. Whenever a new requirement for a different type of agent arises, programmers for Aspen-EE have had to develop specific rules (and hence computer programs) to implement the behavior of the new class of agents.

Agent Classes in Aspen-EE	
Household	*Fuel Company*
Government	*Generation Company*
Commercial	*Independent System Operator*
Industry	*Bulletin Board*
Disaster	*Weather*

In Aspen-EE there were three kinds of markets in which the agents interacted: product, labor, and electric power. A majority of the agent types in Aspen-EE were designed for the electric power market to represent the producers of electricity, the market structures that control the production of electricity, and the suppliers of electric utility requirements.

A New Architecture for CommAspen

For CommAspen, we changed the model structure to an architecture that is much more generic and functional, and that allows us to build different types of agents based on capabilities, or functionality. And, instead of having different types of markets, each with its specific rules, there is a generic market with a set of general rules. Using the inheritance capabilities of C++, we are able to overwrite the general rules for both agents and markets to create specific types of capabilities that are tailored for the type of context and problem being addressed.

Building Agents

In CommAspen, it is really all about plug-and-play. That notion is important in understanding the new architecture. Any type of economic agent in CommAspen is called a *CommAgent* and is composed of a set of building blocks, or templates. There are quite a number of blocks used, each with its own unique name. Of these many blocks, there are a few that represent an agent's basic structure. Figure 1 shows the primary blocks that compose an agent. Model-assigned names are used for these blocks.

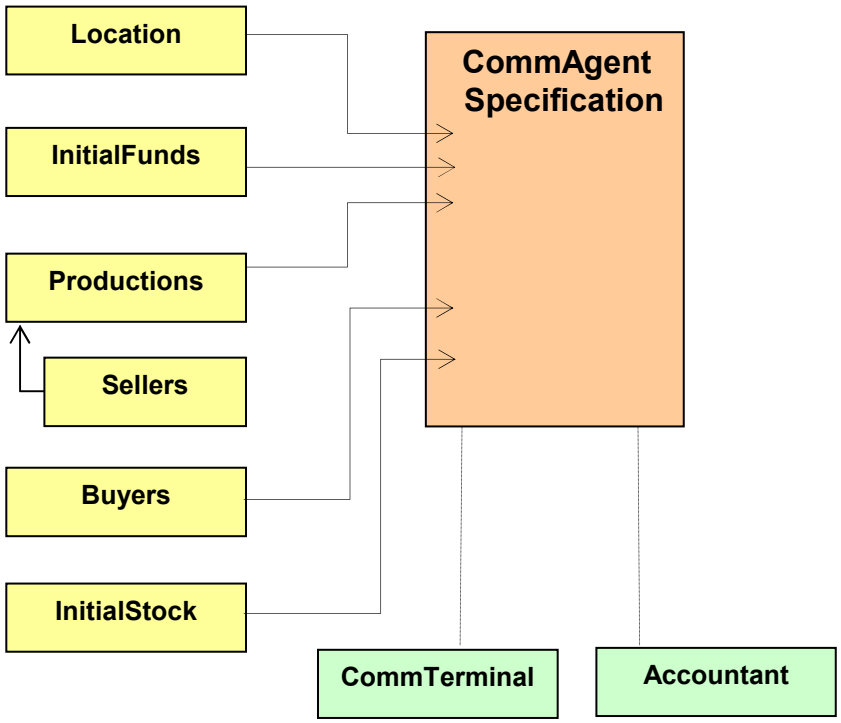


Figure 1. Major blocks used to build a CommAgent.

Users of CommAspen build agents with the six blocks on the left in Figure 1. The "big three" blocks are *Productions*, *Sellers*, and *Buyers*. Essentially, each of our economic agents has one or more productions to drive it. The agent may need to sell its productions

somewhere and will use sellers to do that. The agent may need to obtain raw material for its productions and will use buyers for that activity. Three other building blocks shown in Figure 1 provide start-up information for a simulation. *Location* defines the agent's starting location. *InitialFunds* allow the agent to have a certain amount of cash on hand at the beginning of a simulation. *InitialStock* specifies numerical attributes of the commodities that the agent has available at the start of a simulation. *Note that throughout this report we refer to goods and services as commodities*.

Currently, all agents in CommAspen are automatically configured with a communications terminal, referred to herein as a *CommTerminal*, and an *Accountant*. The CommTerminal enables agents to communicate with one another for financial transactions. The Accountant keeps track of most of the agent's financial records.

With the blocks identified in Figure 1, any number of types of CommAgents can be built. We have used the blocks to construct several types of CommAgents (see inset on right) for the telecommunications and banking problem in this report and for use in the CommAspen library as well:

CommAgent Types in CommAspen

Consumer
Firms
Bank

- The *consumer* agent type consumes commodities and gets a paycheck every so often, but does not work. This is a very specialized type of agent that buys but has nothing to sell.

- *Firm agents* produce commodities for sale to other firms and consumers, and buy commodities for use in their productions. Any number of firm agents can be built with the same set of building blocks.

- The primary purpose of the *bank* agent type is to clear checks for financial transactions that occur during the simulation. Certain functionality has been added for this particular type of CommAgent so that it can specifically handle financial transactions that other types of CommAgents cannot do.

Associated with the basic building blocks of Productions, Sellers, and Buyers are sets of more specialized templates or blocks, with their own unique names, to implement the capabilities for the particular types of agents we have built. Thus, for example, we have three templates for the Productions block, based on the agent's type: the Consumer block, the FirmProduction block, and the BankAccountProduction block. These blocks, and others subordinate to them, are part of the CommAspen library and available for constructing new types of agents for different problems.

Special Agent Types in CommAspen

Router
Bulletin Board

Behind the scenes are special types of agents whose capabilities are incorporated in the computer code:

- The *router* agent type was built specifically for the communications network.

- The *bulletin board* agent type allows other agents to communicate with each other. Markets for commodities are examples of bulletin boards.

Through a few parameters in the input file, users have a limited amount of input into the ways these special types of agents work in a simulation.

Defining Agent-to-Agent Relationships and Markets

It is through the major building blocks that users define not only the relationships within each type of CommAgent but also the relationships across CommAgents of different types. Such relationships define the rules of behavior by which the individual agents of all agent types will interact with each other in a simulation. Regarding intra-agent relationships, those who sell commodities need sellers, those who buy commodities need buyers, and those who make or consume commodities need raw materials to use in their production process. For all practical purposes, consumption is a form of production in CommAspen. Regarding interagent relationships, we need to match those types of agents who sell commodities with those types of agents who buy commodities.

Assume that we have two types of agents, Firm_A that produces and sells Component_A and Firm_B that produces and sells Component_B. To make Component_B, however, Firm_B needs to buy Component_A from someone who sells that component, like Firm_A. Figure 2 shows at a very high level some of the intricacies involved in defining these internal and external types of relationships.

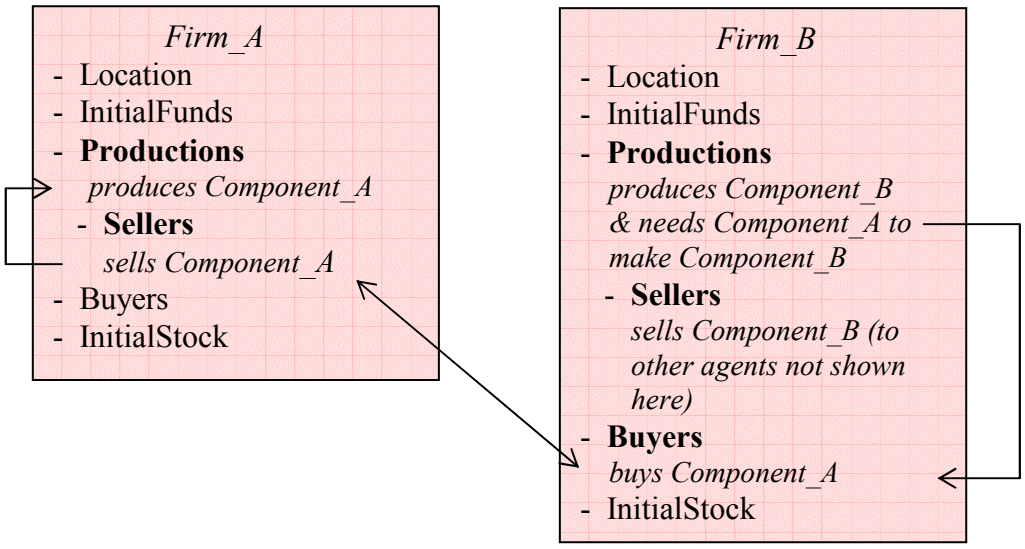


Figure 2. High-level view of some agent-type interrelationships and intrarelationships.

What the agent produces is identified in its Productions block, which is associated with its own Sellers block. Also any ingredients needed to produce its output is defined as part of the Productions block, i.e., Firm_B needs Component_A to make Component_B. And any ingredient that is specified in a Productions block must have a corresponding

buyer from which to purchase the particular commodity. An intra-agent relationship between buyer and seller is thus defined.

Accordingly, through the Productions, Sellers, and Buyers blocks that are common across all agent types, we define the interrelationships between different types of agents and within agents of the same type. As the interrelationships between different agent types are defined, markets for the particular commodities are thus created in CommAspen. The term "market" can be thought of most easily as the context in which a particular commodity is bought and sold. From that perspective, a market is not a store or a real place to buy something, but rather a conceptual relationship between various entities. Some of the entities want to sell things; others want to buy things. Collectively, these entities, i.e., agents in CommAspen, make up the market for a particular commodity, with the sellers competing for business. Conceptually, a market is a big bulletin board where everyone in the market can, simultaneously, find out who else is in the market, what they are charging, and how much of the market is dominated by any individual seller.

To be able to participate in a market, agents have to be able to follow the rules of the market. Each market can have different rules. We encapsulate the rules in a buyer/seller pair, where the buyer side knows "how to buy" and the seller side knows "how to sell." The market then indicates which agents know how to sell, and hence which agents another agent can buy from using its corresponding buyer. The market does not indicate what types of agents are participating—only that they know how to follow the rules. Figure 3 shows an example of how markets are created.

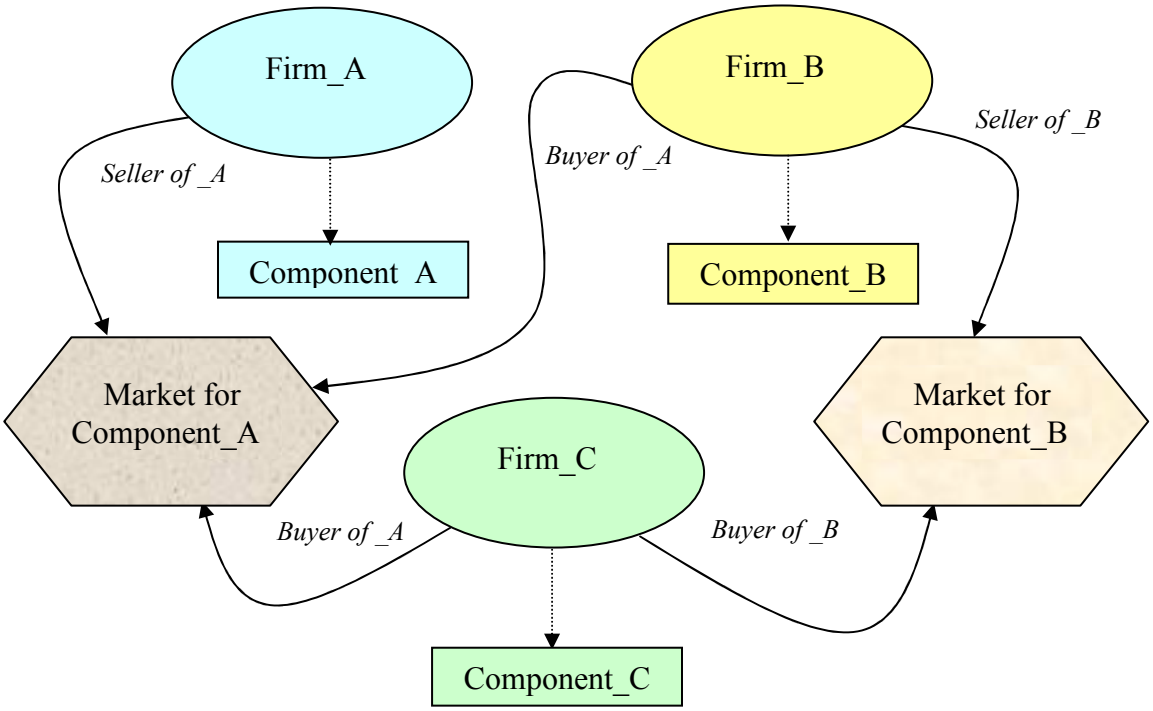


Figure 3. Example of creating markets.

In Figure 3, we have identified two markets: one for Component_A and another for Component_B. There would be another market for Component_C, but that market is not shown.

- Firm_A produces Component_A and sells it in the Market for Component_A. Both Firm_B and Firm_C are buyers in the Market for Component_A.

- Firm_B produces Component_B and sells it in the Market for Component_B. Firm_C is the only buyer in the Market for Component_B.

Of course, in our simulation we would have many instances of individual firms of these agent types, and the exchange behaviors of these individual agents would quickly become complex. For the above example, markets were created for generic commodities. But we could just as easily have created markets for electric power or labor or other types of commodities in this same fashion.

CommAspen agents and markets are subject to external conditions, like communications outages, and to internal decisions within markets, such as inability to fill sales orders by sellers.

Representing Infrastructures

There are several ways that we can implement the notion of infrastructures in CommAspen. One method of representing certain types of infrastructures in CommAspen is through the use of *spigots* and *sinks*. Such infrastructures are for commodities that run continuously, like water from a municipality and electricity from a local utility. A sink is where a producer puts product into an infrastructure. For example, a power company may have a natural gas-fired electric generating plant producing power. It would put power on the transmission lines by passing the power into the associated sink. A spigot is where a consumer gets the product, such as turning on the lights in a residence or getting water from a faucet.

The actual creation of sinks and spigots is hidden from the user. An agent that buys power, for example, from a power market, automatically gets a spigot from which it can draw the power. An agent that buys water from a bottled water company, however, gets shipments of bottled water, not a spigot.

Sinks and spigots serve to communicate demand and availability between a producer and a consumer. Relative to electric power, for example, the consumer agent is restricted by a power outage since they will not be able to make purchases electronically during an outage. Similarly, it is important for the producer to know that the cumulative pull on the power system at a given moment is greater than its ability to produce, so that the CommAspen program can know when to "start" the outage.

Another method for representing infrastructures is through the construction of a network; such as we have done with the communications network in CommAspen. If

communications systems are overwhelmed with the sheer volume of communication, they could ripple through the system and, in effect, cause an outage.

Summarizing Differences Between Aspen-EE and CommAspen

While CommAspen is built upon the Aspen-EE model, CommAspen differs in several important respects. First, as discussed previously, CommAspen has a very different architecture and thus way of building agents. This has implications for user inputs. The set of input parameters is actually much smaller in CommAspen than in Aspen-EE. Second, the agent classes in CommAspen are more general than those found in Aspen-EE. All of the Aspen-EE agents as well as those that were present in Aspen, the first of the model series, can be developed from the general CommAspen components. Third, methods for implementing markets and infrastructures differ between the two models. Fourth, the representation of the economy in CommAspen is more simplified than that found in Aspen-EE. Importantly, however, Aspen and Aspen-EE agents can easily be modified and used in CommAspen for user problems in which the government, for example, is of interest. For a description of agents in their original form, see *Aspen: A Microsimulation Model of the Economy* [6]. Fifth, CommAspen features a communications network not found in Aspen-EE.

The Mechanics of CommAspen

Agents in CommAspen, like in previous Aspen models, are decision makers. Each agent behaves the way its counterpart in the real world would behave, as the simulation traces the agent's daily actions, e.g., buying commodities, selling commodities, paying for commodities by check, credit card, or other electronic payment means, etc. Agents of the same type draw from the same decision rules. For example, all firm agents of the same type, e.g., Firm_B, use the same rule to decide from which agent that they will purchase commodities. However, the decision from which agent to actually purchase a commodity may vary from agent to agent because of its own constraints, such as insufficient income to purchase the commodity desired at a particular point in the simulation or because an agent is more concerned about getting the lowest price and spends more time shopping.

As explained previously, the user builds agent types with a set of building blocks, or templates, each with its own name. The number of individual agents of each type created during a simulation is also specified by the user. For example, a simulation in which the user creates two types of firms, Firm_A and Firm_B, might contain 10 individual Firm_A agents and 20 individual Firm_B agents.

A simulation in CommAspen is a sequence of *events*. Fundamentally, an event is just a point in the sequence where something "interesting" happens. Therefore, an event is not an *action*—an event is more like a piece of paper from a "take-a-number" paper dispenser that one might find at the Motor Vehicle Division (MVD). That is, an event gives a priority to an abstract concept so that it can be sorted and scheduled. To use the MVD example, your business (which is perhaps a driver's license renewal) is an abstract

concept that can be sorted and ordered simply by giving you a piece of paper with a number on it. The order in which MVD customers are processed is then given by an ascending sequence (1, 2, 3, …). That is to say, as a customer, you have been given a numerical priority that is independent of your business and one that roughly corresponds to the order in which you arrived at the MVD office (that is, it's a "fair" priority for the particular situation). In CommAspen the prioritization scheme is a bit more complicated (i.e., we also prioritize by your "business"), but the concept is the same. The simulation is accomplished by processing the events in a correct sequence.

There are several types of events; all have important roles in producing the final output of the model. Two kinds of events, however, do most of the work: task events and message events. Both event types involve model computations. A task event is one in which an agent independently begins a new sequence of actions. A message event is one in which an agent communicates with another agent, possibly to complete a sequence of actions. Task events have higher priority than message events.

A task event usually starts off a chain of message events. For example, an agent can decide to pay all its bills on the first of the month. So every month, it schedules a task, "Pay All Bills" to remind itself to pay its bills the next month. Then, when the time comes the next month to "Pay All Bills," the agent sends off a series of messages along the lines of "I am paying my bill; here is a check for $50." These messages are scheduled on the calendar (see below) for the moment when they should arrive, say, four days in the future for a check sent through the postal system—and the moment they arrive is triggered by a message [delivery] event scheduled specially for that purpose.

CommAspen uses the concept of a calendar to sequence events during a simulation. Events are scheduled on the calendar (a priority queue), with different priorities assigned by CommAspen programmers to different events. Priority is determined by the event's time, its type, and any applicable secondary priority. At the top of the calendar, which changes dynamically, is either the highest priority event, or one of several events that have equivalent priority, where that priority is the highest priority relative to all other priorities of events on the calendar.

Time in Aspen-EE is divided into a minimum time step that is determined by CommAspen users. Thus, one time step can be set to be equivalent to one minute, and any other unit of time can be derived from that. For example, if the minimum time step was one minute, as it is in the current version of CommAspen, an hour would take 60 time steps, a day would take 1,440 time steps, and a week would take 10,080 time steps. Each time step can have zero or more events. Since events are prioritized by time, type, and secondary priority, one can interpret the priority scheme as "dividing" the time step into stages. For example, all task events have higher priority than any message event; hence, all task events will be processed before the first message event.

CommAspen Input and Output

The model reads a single user-prepared text file as input and can produce several kinds of output files. General characteristics of these files are discussed below.

Input File

The CommAspen Input File provides initial values for a number of parameters needed during the simulation. Each input file is a separate problem. The format of the CommAspen Input File differs from that found in previous Aspen models. This new format is prepared in XML, following an organization defined by CommAspen programmers. As described in Appendix A, the file contains three sections. The first section specifies parameters related to the XML document itself. The second section is generally reserved for comments about the run or the problem. The third section, the major and largest part of the input file, gives characteristics of the run and of agent types that will be created for the problem. The number and names of agents of the particular types are also specified.

Initial values for characteristics of the agent types are required because many of the decisions in CommAspen depend on the current state of those characteristics in specific individual agents. For example, an agent's consumption decisions depend on factors such as its ability to pay for commodities it purchases. Therefore, at the beginning of a simulation, each agent is assigned an amount of starting cash , i.e., as input in the InitialFunds block. A starting value for this kind of parameter is typically specified by the user as a constant (e.g., amount = '1000') or as a range of values from which a single value is chosen (e.g., amount = '1000 – 5000'). Generally, any numerical input can be entered as a constant or range. When expressed as a range, chosen values for all the individual agents of the specific type are uniformly and randomly distributed over that range. By using this approach, CommAspen can usually create agents that are "different" enough to represent a heterogeneous population. Sometimes, however, this approach will not generate a satisfactory distribution, and an actual distribution can be specified instead. For example, CommAspen employs an actual distribution for use behaviors related to consumption of commodities such as power. To accommodate the specification of actual distributions and other kinds of ordered data, CommAspen accepts an *array* of initial values for certain input parameters. In the case of consumption, there is an array of labeled hours' fields, each permitting the specification of an amount used per hour.

During a simulation, the initial values assigned to some of the parameters in the input file can and do change. For example, the amount of starting cash assigned to an agent will change from its initial value as the agent spends money for commodities. On the other hand, certain initial values do not change, e.g., the maximum acceptable price that an agent will pay for a commodity (as specified by max_acceptable_price in the Buyers block).

Appendix A presents the format of the input file and provides a tabular description of the set of input parameters. Examples of initial values assigned to these parameters are

given. Note that if the user does not enter an initial value for a parameter, a default value is used, and the user is warned to that effect. Appendix B gives a sample input file used to run the problem investigated in this report.

Note that as new types of agents are created for new problems, new templates, i.e., agent building blocks, will become available in the CommAspen library for use in other problems. Additionally, as new functionality is added to CommAspen, the set of input parameters will be increased or modified as necessary.

Output Files

CommAspen produces a primary data file called the Snapshot File from which other types of files can be derived by condensing or otherwise manipulating the snapshot data itself. The Snapshot File contains a subset of the model's state at periodic intervals, notably the state or values of interesting information like agents' current cash assets. As the simulation progresses, the model sends the Snapshot File to a network server called SimStreamer. This server accepts connections from clients running on Windows and effectively relays the data found in that file to the connected clients. Thus, users can watch the simulation as it is running. The Snapshot File can also be exported into a text file that, in turn, can be read by programs such as Microsoft Excel™ or Matlab™ to produce graphic output like that shown in the **Results and Analysis** section of this report.

CommAgent Blocks and Their Rules of Behavior

CommAgents can buy, sell, consume, and produce commodities. They also can use the communications network in the process of conducting financial transactions. Once per time step, a CommAgent directs its Productions block to produce, or in some cases just consume. Production levels are adjusted as commodities are used.

The capabilities of a particular type of CommAgent are defined in the input file using the individual building blocks presented in this section. Readers will note that several of the basic building blocks offer a variety of specialized capabilities for building agents. These capabilities translate to additional building blocks within the framework of the major building blocks. Figure 4 presents the basic structure of a CommAgent in the input file.

```
<ObjGen name='user-defined'>
  <CommAgent>
    <Location
    </Location

    <InitialFunds>
    </InitialFunds>

    <Productions>
    </Productions>

    <Buyers>
    </Buyers>

    <InitialStock>
    </InitialStock>

  </CommAgent>
</ObjGen>
```

Figure 4. Basic structure of a CommAgent in input file.

A CommAgent specification in the input file is made within an ObjGen block. The block begins with a unique name that the user assigns to the agent type (e.g., <ObjGen name='Firm_A'). That name is then followed by a block of the form <CommAgent> … </CommAgent>, which contains the building blocks that characterize the agent type.

Each of the major blocks listed in Figure 4 should be present in a CommAgent specification for any type of agent if the block contains any user-input items. Otherwise, the block may be omitted. To facilitate understanding of the blocks, we have used snippets from several sample input files that show examples and usage of the input data. Some of these snippets were taken from the sample input file in Appendix B, but not all.

Location

Input File example:

```
<Location
      latitude='0 - 10000'
      longitude='0 - 10000'
      elevation='0.0'/>
```

Every CommAgent has a starting location, specified in terms of latitude, longitude, and elevation. Within a Location block, the user can specify a range for each of these parameters, which allows the model during the run to randomly place the individual agents of the particular agent type within the specified ranges.

Two reasons motivated inclusion of location for an agent. First, these data support the transportation of goods, a future enhancement for the model. Second, location enables us to group different agent types, like firms and consumers, into different regions so that they can buy, if so desired by the user, from their own region only. By observing local markets interacting with one another, we can capture a kind of economic behavior that was not possible in previous Aspen models.

InitialFunds

Input File example:

```
<InitialFunds amount='1000000'/>
```

Every CommAgent begins the simulation with a certain amount of cash, as specified in the InitialFunds block. As the simulation progresses, these funds can increase and decrease depending on the kind of buying and selling activities in which the agent is engaged.

Productions

To participate in any kind of market during the simulation, a CommAgent needs a Productions block. This block allows the agent to produce and sell commodities, and also to consume commodities. Informally, a production is something that takes other commodities as input and produces as output a new commodity or service. This definition also covers the relabeling of production, where inputs could be boxes of loose bolts, paper, plastic, power and labor, and the output could just be little bags of bolts suitable for retail sale.

For each commodity that is part of the agent's production, there must be one specialized block, referred to herein as a *type-of-production* block, in the Productions block that performs the production activity. The specialized type-of-production blocks are selected from available templates. Currently, model users can use the templates named "FirmProduction," "Consumer," and "BankAccountProduction." The FirmProduction block has been used to define all six types of firm agents in our sample problem. The Consumer block covers only the consumption function and can be used to describe the bank agent type and also the consumer agent type. The BankAccountProduction block has been used solely for the bank type of agent in our sample problem.

The specific contents of a type-of-production block differ, based on the type of agent being constructed. Again, there are additional building blocks for describing particular capabilities that the particular type-of-production block has available. For reference in the subsequent discussion, Table 1 identifies the valid names of the blocks used in the three type-of-production blocks.

Table 1. Type-of-Production Block Contents for Specific Agent Types

All Firm Agent Types	Consumer	Bank
<Productions> <FirmProduction> <Sellers> <FirmSeller> <INPUTS> <INPUT>	<Productions> <Consumer> <DAILY_CONSUMPTION> <CONSUME> <DISTRIBUTION>	<Productions> <BankAccountProduction> <DAILY_CONSUMPTION> <CONSUME> <DISTRIBUTION> <Sellers> <BankAccountSeller>

A CommAgent can have more than one type-of-production block. Multiple productions might be useful, for example, in supply-chain problems, where the ultimate commodity produced was composed of subassemblies. A user could also define a type of agent that both produced some output commodity and consumed other input commodities.

In the paragraphs below, we explain by example how agents are constructed with the three type-of-production blocks.

Example One: Defining a FirmProduction Block

A FirmProduction block is currently used to define a production for those types of firm agents in our sample problem that are involved in producing some output. The production itself does not sell anything; instead, it produces the stuff and "disposes" of it in the sense that it puts it in the warehouse or down a sink. However, the seller needs to tie into the production to adjust the production levels. For example, a production line may lie idle until the seller gets an order, in which case the production line fires up long enough to fill the order. (This is how a small bottling company may operate—the company may keep a small amount of stock on hand, but when the big order comes in, the company runs the production for a few days to fill the order and builds up a small cushion in the warehouse.) On the other hand, the small company may have a huge warehouse that is filled with the fruit of past production, e.g., boxes and boxes of stuff. In that case, the company can sell some stuff and cook the books, but meanwhile not actually use up any input to produce its output, i.e., the production is dormant, but its output is (still) being sold.

For any commodity, three initial parameters need to be specified by the user: (1) the initial quantity of the commodity that will be produced daily, (2) the maximum quantity of the commodity that will be produced daily, and (3) the name of the commodity (i.e., OUTPUT=). In the example below, the agent produces COMPONENT_A. This commodity is sold to other agents in the simulation via the associated Sellers block specification, as will be discussed next.

Input File example:

```
<Productions>
    <FirmProduction initial_amount_produced_daily='500'
                    max_amount_produced_daily='675'
                    OUTPUT='COMPONENT_A'>
      <Sellers>
        <FirmSeller
         region='1'
         initial_advertised_price='1.17 - 3.00'
         sampling_prob='0.3333'/>
      </Sellers>
    </FirmProduction>
</Productions>
```

Identifying Sellers of the FirmProduction Block and Their Behavior

For each FirmProduction block specified in the input file, there must be a corresponding and embedded Sellers block. The Sellers block provides market-specific information about the particular commodity specified in its associated FirmProduction block. As with other CommAgent components, a specialized *type-of-seller* block has been created for the FirmProduction block: FirmSeller. Three input parameters can be specified for a FirmSeller block: (1) the region, or market, in which the commodity can be sold; (2) the initial price of that commodity; and (3) the amount of presence that the agent will have in the particular market (via the sampling_prob parameter).

There can be one or more FirmSeller blocks in a Sellers block. However, a different region or market must be specified for each such FirmSeller. What this means is that an agent can sell the same commodity in different markets. Here is an example in which COMPONENT_A would be sold by a particular type of agent in two different markets, i.e., region 1 and region 2. Note that the advertised price and sampling probability are the same, but they could be different.

Input File example:

```
<Productions>
    <FirmProduction initial_amount_produced_daily='500'
                    max_amount_produced_daily='675'
                    OUTPUT='COMPONENT_A'>
      <Sellers>
        <FirmSeller
         region='1'
         initial_advertised_price='1.17 - 3.00'
         sampling_prob='0.3333'/>
        <FirmSeller
         region='2'
         initial_advertised_price='1.17 - 3.00'
         sampling_prob='0.3333'/>
      </Sellers>

    </FirmProduction>
</Productions>
```

Note that any number of agent types can produce and sell the same commodity in a simulation, thus creating a competitive marketplace for each such product.

The logic of buying and selling is paired between sellers in one agent and buyers in another agent. Thus, in order to create a market for COMPONENT_A, at least one other agent type in the simulation would need a corresponding FirmBuyer for COMPONENT_A. This type of pairing is discussed under the **Buyers** subsection of this discussion.

The following paragraphs describe particular behaviors related to how prices are determined for commodities and how market presence is implemented.

Commodity Sales and Prices. Agents will sell a certain amount of products per day. To simulate how the agents, like firms, set prices for their commodities, CommAspen uses a genetic algorithm learning classifier system (GALCS) in which the agents determine four trends daily: (a) whether the commodity price has been recently increasing or decreasing, (b) whether sales have been recently increasing or decreasing, (c) whether profits have been recently increasing or decreasing, and (d) whether prices are higher or lower than the industry average. Based on answers to (a) through (d), the agent finds itself in one of 16 states.

The GALCS assigns a probability vector (p^D, p^I, p^C) to each state,

where p^D = probability that the agent will decrease a given price (by a certain exogenously specified amount) the next time the agent enters the same state,
p^I = probability that the agent will increase the price, and
p^C = probability that the agent will keep the price constant.

Upon entering a certain state, the agent decides how to change a given price by using the corresponding probability vector and choosing a random number. The agent then adjusts

the vector according to how the price change affects profits. The example below can help to explain this process.

Suppose that at a particular time for state 2, the following condition exists: $(p^D, p^I, p^C) = (0.1, 0.6, 0.3)$. Assume that an agent then enters this state and draws a random number indicating the need for a price increase. Suppose further that as a result of increasing the price, profits drop. To reflect this drop, the vector is then adjusted to (0.15, 0.5, 0.35). Thus, CommAspen simulates the agent's learning process. The agent learns that raising prices in state 2 was detrimental. As a result of an incorrect decision, the vector is adjusted to reflect a decreased probability of a price increase. The changed probability vector reflects the unlikelihood that the agent will increase prices upon re-entry into state 2.

For more details on GALCSs and GALCS results from other Sandia runs, see *Aspen: A Microsimulation Model of the Economy* [6].

Market Presence. CommAspen has introduced the notion of market presence as an attribute of a FirmSeller. The logic for market presence is implemented as a sampling probability through the input parameter named sampling_prob. The higher the presence of an agent in the market (whether by advertising, local sensibility, number of locations, or any other reason), the more other agents are likely to be aware of that agent and to sample the prices of its commodities. The value of the parameter sampling_prob is best set by the user in consideration of all agent types selling the same commodity in the simulation. For example, if three types of firms sell COMPONENT_C and all have sampling_prob equal to 0.3333 (33%), they will have an equivalent market presence. To differentiate sellers by market presence, the value of this input parameter should be different. Ideally, the sampling probabilities will sum to 1.0, but if they do not, the program will normalize the values so that they do sum to 1.0.

The effect of having a higher market presence is that one's prices have higher visibility on the bulletin board. See the section titled **Bulletin Board Agent Rules** for further details.

Specifying Production Recipes for the FirmProduction Block

What distinguishes different types of firm agents, most directly, from one another is the recipe they use to produce their output. Recipes are a new concept and implementation for the Aspen models. In CommAspen, a recipe specifies the input ingredients necessary to produce a unit of output. Inputs to a recipe can include labor, infrastructure commodities (like power and water), and goods.

Inputs to a production are specified through the INPUTS block.

Input File example:

```
<Productions>
    <FirmProduction initial_amount_produced_daily='600'
                    max_amount_produced_daily='650'
                    OUTPUT='COMPONENT_C'>
      <Sellers>
        ...
      </Sellers>
      <INPUTS>
        <INPUT commodity = 'COMPONENT_A'  amount = '1'/>
        <INPUT commodity = 'COMPONENT_B'  amount = '1'/>
      </INPUTS>
    </FirmProduction>
</Productions>
```

In the example above, the agent is producing one commodity, expressed through the OUTPUT parameter. The INPUTS block, composed of two INPUT items, is used to specify the ingredients, COMPONENT_A and COMPONENT_B, that are needed to produce COMPONENT_C. For each INPUT item, the name of the ingredient (commodity) and the quantity of that ingredient (amount) are specified. Each such commodity will need to be purchased by a buyer belonging to the agent type.

Note that in a production recipe, the output is not the sum of the input (i.e., 1.0 unit of milk and 1.0 unit of chocolate syrup do not result in 2.0 units of chocolate milk. Instead, the formula may read: 1 unit of chocolate milk is produced by 1.0 unit of milk and 0.02 unit of chocolate syrup. The recipe specification assumes 1 unit of output—so to make 1 unit of output requires all of the listed inputs in the amount given. [Since the units in the input table roughly correspond to dollars, the difference between $1 of output and the cost of inputs is the profit margin; the smaller the sum of inputs, the more profitable the output.]

Example Two: Defining a Consumer Block

Some types of agents may consume commodities like our consumer agent in the sample telecommunications and banking problem. For this purpose we have another type-of-production block, named Consumer, that tells the agent what it needs. That need is expressed through a DAILY_CONSUMPTION block that lists the commodities that will be consumed by the agent. And associated with the DAILY_CONSUMPTION block is another block named DISTRIBUTION that specifies the schedule, or use behavior, by which the commodities listed will be consumed by the agent. The current version of CommAspen allows only one DAILY_CONSUMPTION block in a Consumer block.

Input File example:

```
<Productions>
   <Consumer>
      <DAILY_CONSUMPTION>
        <CONSUME amount = '5-10' commodity = 'COMPONENT_F' />
        <CONSUME amount = '1-5' commodity = 'COMPONENT_E' />
      </DAILY_CONSUMPTION>
      <Distribution
       hour0='0.0' hour1='0.0' hour2='0.0' hour3='0.0'
       hour4='0.0' hour5='0.0' hour6='0.0' hour7='0.0'
       hour8='1.0' hour9='1.0' hour10='1.0' hour11='1.0'
       hour12='1.0' hour13='1.0' hour14='1.0' hour15='1.0'
       hour16='0.0' hour17='0.0' hour18='0.0' hour19='0.0'
       hour20='0.0' hour21='0.0' hour22='0.0' hour23='0.0' />
    </Consumer>
</Productions>
```

Tip: The particular consumption for our consumer agent in the sample problem in this report represents a family size of one. By varying consumption amounts, one could create different types of consumer agents and give them names like "Consumer_2members", "Consumer_4members", and so on.

Specifying Commodities and Their Use Behaviors

The DAILY_CONSUMPTION block can contain one or more CONSUME items. For each CONSUME item, the quantity of the commodity needed (amount) and its name (commodity) are specified. Each such commodity will need to be purchased by a buyer belonging to the agent type.

The DISTRIBUTION block contains a set of 24 entries of the form "hour*n*= *value*" (*n* is the specific hour, 0 through 23), with the amount consumed of all such commodities listed in the associated DAILY_CONSUMPTION block specified on an hourly basis. Each normalized hourly value represents the fraction of the total daily use actually used within the hour. Currently, the schedule defaults to the amount the agent consumes from 8 A.M. to 4 P.M. This schedule can be changed in the input file. Note that the type of production referred to as FirmProduction also has this type of distribution, but it is hard-coded and does not currently accept user inputs.

The consumer agent in our sample problem (and in the CommAspen library for use in other problems) is a production in disguise, so its daily consumption corresponds to a production recipe needed to produce one unit of a throwaway commodity over a 24-hour period.

Example Three: Defining a BankAccountProduction Block

A BankAccountProduction block is used to define the production for a bank type of agent only. Currently, this type of production has no defined input parameters and is considered a null or "free production." Further development of this special type of agent

in CommAspen is anticipated to the point where its production capabilities will be more like those found in the FirmProduction block.

Input File example:

```
<Productions>
   <BankAccountProduction>
     <Sellers>
       <BankAccountSeller region='1'/>
     </Sellers>
   </BankAccountProduction>
 </Productions>
```

Specifying Commodities and Their Use Behaviors

While not shown in the example above, a bank type of agent also allows specification of a DAILY_CONSUMPTION block and its associated DISTRIBUTION block.

Identifying Sellers of the BankAccountProduction and Their Behavior

For each BankAccountProduction block specified in the input file, there must be a corresponding and embedded Sellers block. The Sellers block provides market-specific information. Similar to the FirmProduction block, a specialized *type-of-seller* block has been created for the BankAccountProduction block: BankAccountSeller. Only one parameter can currently be specified for this block: the region, or market, in which the banking activity will be conducted. There can be one or more BankAccountSeller blocks in a Sellers block. However, a different region or market must be specified for each such BankAccountSeller.

The logic of buying and selling bank accounts is paired between sellers in the bank type of agent and buyers in agents of other types (and also the bank agent itself). All CommAgents require a bank account so that they can pay their bills and cash their checks. Thus, you will notice that all of the agent types in our sample problem have a BankAccountBuyer, but only the bank type of agent has a BankAccountSeller.

Buyers

Any CommAgent with a Buyers block can purchase commodities from other agents who sell those commodities. A Buyers block tells the agent how to go about obtaining what it needs. Essentially, then, buyers are a way to obtain raw materials for the production.

Buyers will purchase some number of commodities each day. For each such commodity, there must be one specialized block, i.e., a *type-of-buyer* block, that performs the buying function. The specialized type-of-buyer blocks are selected from available templates, which currently are “BankAccountBuyer” and “FirmBuyer”. Every type of agent must have a BankAccountBuyer block if it is to participate in financial transactions during a simulation. The presence of FirmBuyer blocks is dependent on whether

commodities have been specified as necessary for production or consumption in other parts of the Productions block for the agent type.

In the paragraphs below, we explain by example how to construct the two different kinds of type-of-buyer blocks.

Example One: Defining a BankAccountBuyer Block

A BankAccountBuyer block allows only a single parameter: the region, or market, in which the banking activity will be conducted. Only one BankAccountBuyer is allowed per Buyers block.

Input File example:

```
<Buyers>
   <BankAccountBuyer region='1'/>
</Buyers>
```

Pairing BanKAccountBuyer Blocks with BankAccountSeller Blocks

For each BankAccountBuyer in one type of agent, there must be a corresponding type-of-seller in one other agent type, e.g., BankAccountBuyer matches to BankAccountSeller, that sells the same commodity. Currently, only the bank agent type has a BankAccountSeller block, and all agent types are configured with a BankAccountBuyer block.

Example Two: Defining a FirmBuyer Block

A Buyers block can have zero or more FirmBuyer blocks, in addition to a BankAccountBuyer block. In the example below, the Buyers block contains one BankAccountBuyer block and two FirmBuyer blocks.

Input File example:

```
<Buyers>
  <BankAccountBuyer region='1'/>
  <FirmBuyer
      region='1'
      commodity='COMPONENT_A'
      order_chunk='1.0'
      max_acceptable_price='10.0'
      search_cost='5.00' />
  <FirmBuyer
      region='1'
      commodity='COMPONENT_B'
      order_chunk='1.0'
      max_acceptable_price='10.0'
      search_cost='5.00' />
</Buyers>
```

For each FirmBuyer block, parameters are available to specify the locale in which the commodity is purchased (region), the name of the commodity (commodity), the quantity of the commodity that needs to be ordered at any one time (order_chunk), and the maximum acceptable price the buyer is willing to pay for the commodity (max_acceptable_price). While agents are generally looking for the lowest price, they do have some loyalty as do people in the real world. For example, some agents may prefer buying from firms close to them, while others may prefer buying from firms they have done business with before. Time-to-shop is also an important consideration. The search_cost parameter allows us to consider unique buying behaviors. A high search cost for a particular product implies that an agent will not spend much time and effort to find the lowest cost.

A buyer can purchase a particular commodity only in one region. Thus for each commodity needed (for consumption or production), there can only be one FirmBuyer specified per agent type. In the example above, there are two FirmBuyers. One is purchasing COMPONENT_A in region 1, and the other is purchasing COMPONENT_B in region 1. A separate FirmBuyer would need to be added to purchase in region 2, for example.

Determining How Many FirmBuyer Blocks Are Needed

A general rule of the Buyers block, in which the FirmBuyer blocks are placed, is that there must be a buyer for each ingredient that the agent needs for its production. Ingredients are specified in two ways: (1) as a production recipe through INPUT items in the INPUTS block embedded in the FirmProduction block and (2) as consumption through CONSUME items in the DAILY_CONSUMPTION block embedded in the Consumer block.

Following upon the general rule stated above, there needs to be one FirmBuyer block in a Buyers block for each ingredient that the agent needs. Assume that we have a production recipe that calls for COMPONENT_A and COMPONENT_ B in order to make COMPONENT_C. The needs for COMPONENT_A and COMPONENT_B would be specified via INPUT items in the INPUTS block. The final commodity produced from these two components would be specified in the OUPUT parameter (OUTPUT= ‘COMPONENT_C’. Consequently, the agent would need two FirmBuyer blocks: one to purchase COMPONENT_A and the other to purchase COMPONENT_B. These interrelationships between components of a CommAgent specification are shown in Figure 5.

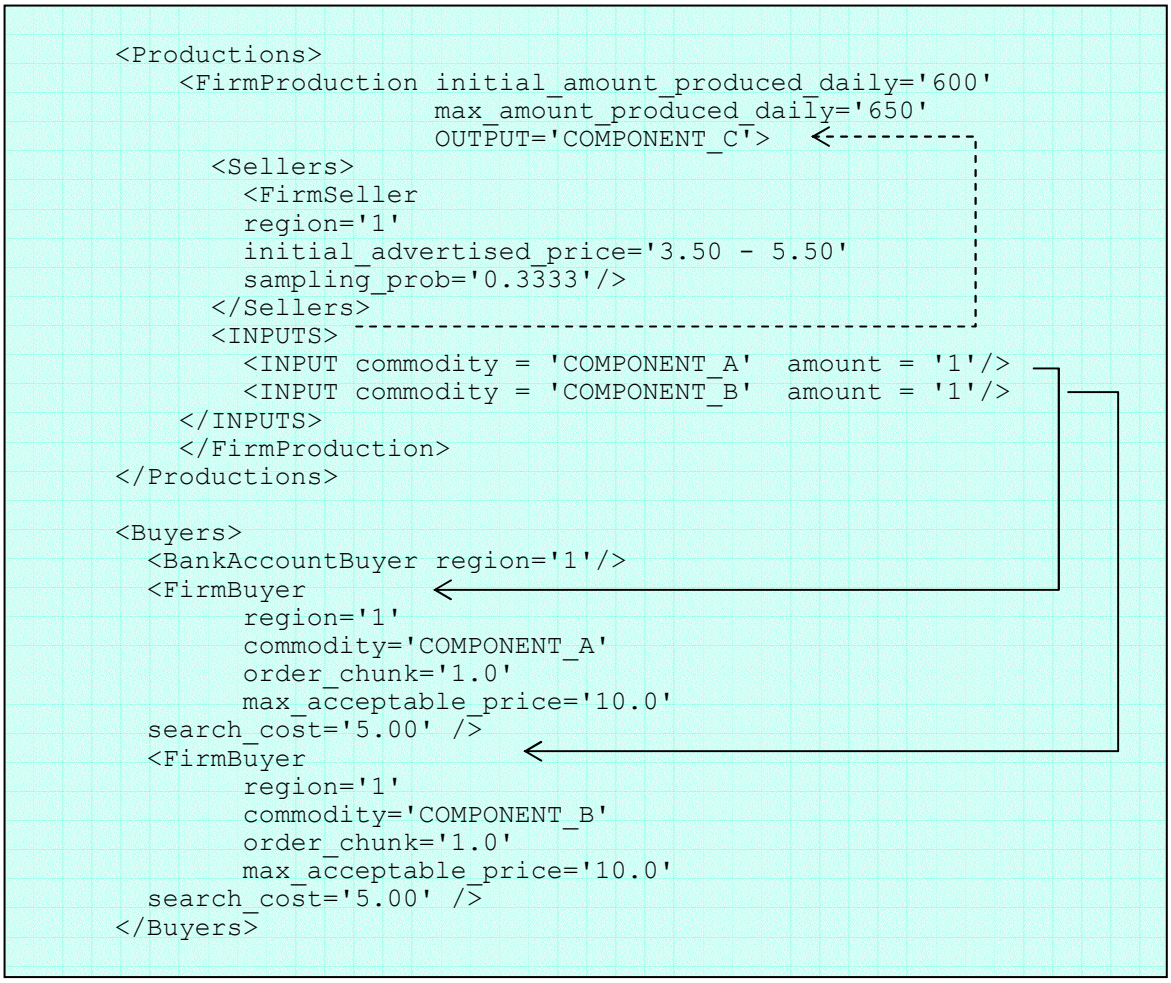


Figure 5. Matching recipe ingredients to buyers.

Pairing FirmBuyer Blocks with FirmSeller Blocks

For each FirmBuyer in one type of agent, there must be a corresponding type-of-seller in at least one other agent type, e.g., FirmBuyer matches to FirmSeller, that sells the same commodity. When a FirmBuyer is paired with a FirmSeller in this way, a market for a particular commodity is created. See the section entitled **Highlights of Interrelationships Across Agent Types** for further details.

Purchasing Commodities

The logic of purchasing a commodity is paired between FirmBuyer and FirmSeller blocks in different types of agents:

1. At the start of the day, a FirmBuyer (of one agent type) determines how much it needs to buy of a particular commodity.

2. To determine whom it will buy from for each commodity needed, the FirmBuyer consults a list of per-unit commodity prices that are broadcast daily by FirmSellers on the bulletin board. The entire list of prices is then passed to a search-cost

algorithm, along with the FirmBuyer's own search cost. The algorithm selects one of the vendors (using random numbers in the sampling process) and returns the Agent ID of the chosen vendor. For a detailed discussion of the search-cost algorithm used in CommAspen, see Reference 7.

3. Next, via the message-passing capability in CommAspen, the FirmBuyer sends an order to the chosen vendor for a certain amount of the desired commodity. The FirmSeller (of another agent type) replies to the order. What happens next depends on the status of the commodity held by the FirmSeller:

    - If the FirmSeller has the needed amount of the commodity in stock, it replies "yes" and ships the order.

    - If the FirmSeller has less than the needed amount of the commodity in stock, it replies "no" and indicates the amount of the desired commodity it has in stock.

4. The FirmBuyer receives the reply from the FirmSeller. If the order is "yes" (and thus shipped), the FirmBuyer is done for the day. Otherwise, the process continues at step 5.

5. The FirmBuyer then reorders from the same FirmSeller the amount of the commodity that the FirmSeller does have on hand.

6. The FirmBuyer then computes the remaining amount it needs of the commodity and repeats the process at step 2 above.

A FirmBuyer does not reconsider vendors it has already ordered from, and the ordering process takes time. Thus after a number of failed orders, a FirmBuyer will give up ordering for the day.

InitialStock

Input File example:

```
    <InitialStock>
      <Stock commodity='COMPONENT_C' amount='100'
max_capacity='1000' cost_per_unit='5.00'/>
      <Stock commodity='COMPONENT_A' amount='100'
max_capacity='1000' cost_per_unit='5.00'/>
      <Stock commodity='COMPONENT_B' amount='100'
max_capacity='1000' cost_per_unit='5.00'/>
    </InitialStock>
```

All CommAgents can begin the simulation with start-up inventory but are not required to do so. The model warms up faster if agents have commodities on hand, which allows them to start consuming or producing immediately. The InitialStock block uses a

Stock item to specify the characteristics of each on-hand commodity that will be stored in the warehouse. A Stock item identifies the name of the commodity the agent sells or needs as input, the amount of that commodity, the maximum capacity the agent can store of the product in its warehouse, and the average cost of each unit of the commodity. Note that CommAspen does not distinguish between the physical attributes of commodities, for example, whether or not it can be contained in a box. Thus, any commodity can be considered as inventory, including such things as water and power.

The notion of a warehouse is new to the Aspen series of models. Through warehouses that belong to agents, CommAspen performs inventory-control functions. The maximum capacity of a commodity in the warehouse is specified as a user input, but if this parameter is not set, the capacity in the warehouse is infinite. The minimum capacity is a just-in-time type of number that specifies the quantity of the commodity in the warehouse that covers an individual agent's needs while awaiting delivery of replacement product. The minimum capacity fluctuates and is set by the FirmBuyer associated with the type of agent in the FirmBuyer's decision on how much to order. The minimum capacity changes every day.

Note: As mentioned previously, there are two other basic components in a CommAgent specification: an Accountant and a CommTerminal. The Accountant keeps track of most of a Commagent's financial records, including the following: unit costs of input material, average and marginal costs of each commodity, revenues and costs, current solvency status, and all transactions the agent has with the banking sector. The Accountant does not, however, maintain records of sales orders. These are tracked by agents' FirmBuyer and FirmSeller blocks. Because it is part of the communications network and has no user-specified attributes, the CommTerminal is discussed in the section titled **Structure of the Communications Network for Financial Transactions.**

Highlights of Interrelationships Across Agent Types

When constructing agent types via the input file, model users need to be aware of ways in which some of an agent's components, i.e., building blocks, are related to each other, either within the agent type itself or between agents of two different types. Several of these relationships are discussed in greater detail below.

Regions

As noted in the block descriptions, CommAspen provides for the specification of regions. A region is a way of grouping markets, infrastructures, and other items into a common area. Regions are defined exclusively through the region parameter in the specialized Buyer and Seller blocks available, i.e., FirmBuyer, FirmSeller, BankAccountBuyer, and BankAccountSeller.

Regions should be considered as instructive, not restrictive. For example, we could allow a FirmBuyer to participate in one region or more than one region. As with other interagent relationships, however, if buyers will purchase a commodity from a particular region, there must be sellers, e.g., a FirmSeller, who also sell such a commodity in that region.

Particular types of buyers look for particular types of sellers during a simulation. For example, FirmBuyers look for FirmSellers in the market for a specific commodity in the same region. A region will have at most one market for a particular commodity. If a FirmBuyer is shopping for COMPONENT_A in region 1, for example, it does not consider any of the FirmSellers found in the market for COMPONENT_B in region 1.

Commodities

CommAspen currently recognizes a list of commodities. These are predefined values, such as 'COMPONENT_A', 'CAPITAL', and 'GOODS', that are specified as part of user inputs in the various blocks that compose a CommAgent specification. The list is given at the end of Appendix A. CommAspen developers can add values to this list for the problem of interest and then such values will be available to model users.

Buyer/Seller Interagent Relationships

This type of relationship occurs across CommAgents of different types. For example, for each FirmBuyer specified in one agent type, there must be a corresponding FirmSeller in another agent type, with both in the market for the same commodity. This same cross-agent-type relationship also holds for the BankAccountBuyer and the BankAccountSeller. A pairing of buyer and seller is shown in Figure 6.

```
<ObjGen name='FIRM_D'>
  <CommAgent>
...
    <Productions>
        <FirmProduction initial_amount_produced_daily='300'
                        max_amount_produced_daily='325'
                        OUTPUT='COMPONENT_D'>
          <Sellers>
            <FirmSeller
            region='1'
            initial_advertised_price='2.50 - 4.50'
            sampling_prob='0.3333'/>
          </Sellers>
          <INPUTS>
            <INPUT commodity = 'COMPONENT_B'  amount = '1'/>
        </INPUTS>
        </FirmProduction>
    </Productions>

    <Buyers>
      ...
      <FirmBuyer
            region='1'
            commodity='COMPONENT_B'
            order_chunk='1.0'
            max_acceptable_price='10.0'
      search_cost='5.00' />
    </Buyers>



<ObjGen name='FIRM_B'>
. . .
    <Productions>
        <FirmProduction initial_amount_produced_daily='400'
                        max_amount_produced_daily='450'
                        OUTPUT='COMPONENT_B'>
          <Sellers>
            <FirmSeller
            region='1'
            initial_advertised_price='1.25 - 2.00'
            sampling_prob='0.3333'/>
          </Sellers>
        </FirmProduction>
    </Productions>
. . .
  </CommAgent>
</ObjGen
```

Figure 6. Pairing a buyer with a seller.

In the above example, Firm_D through its FirmBuyer will purchase Component_B. A matching FirmSeller for Firm_B, which produces Component_B, enables these two types of firms to participate as buyers and sellers in the market for Component_B.

Bulletin Board Agent Rules

The bulletin board is a special type of agent in CommAspen. This agent serves as a dynamic contact point through which other agents share and retrieve information on the environment. Currently, only the current prices of commodities are posted on a bulletin board. In practice, there are quite a number of individual bulletin boards that are active during a simulation. But these are hidden from the user and known only to the agents who access them. As shown in Figure 7, agents who buy commodities access a bulletin board and agents who sell commodities post prices to the bulletin board. Functionally, a bulletin board diverts some of the message passing that needs to occur between agents and, in effect, speeds up processing time.

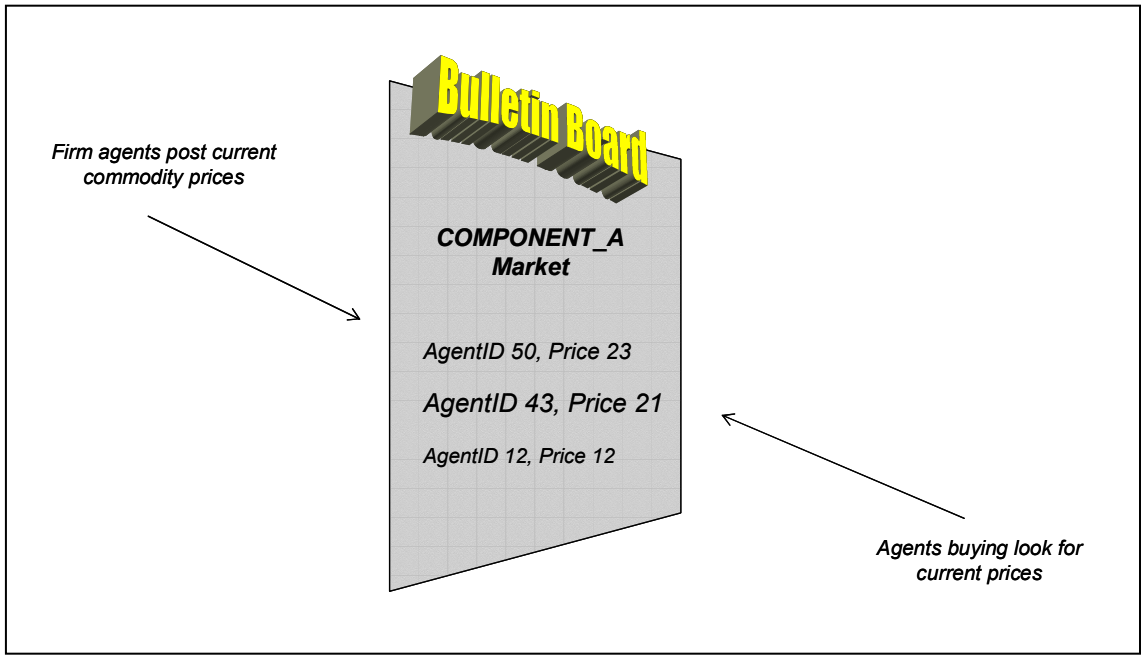


Figure 7. Examples of daily postings on a bulletin board.

Each market is oriented around a single commodity with its own special bulletin board, so the listings of "for sale" prices include only the identifier for the agent (AgentID) and its current price. The commodity listings are sized according to the market presence of the various agents advertising on the bulletin board; the larger the listing, the more it captures a shopper's attention. The higher the sampling probability (a user input specified via the sampling_prob parameter in the Sellers block), the larger the listing.

The sample bulletin board above indicates that the agent with ID 50 (who turns out to be an agent type of Firm_1) is offering COMPONENT_A for $23 a unit and has 33% of the market presence. The table also indicates that other firms, those with IDs 43 and 12 (who turn out to be Firm_2 and Firm_3 agent types, respectively) are also selling COMPONENT_A. The agent with ID 43 has the largest market presence of the three entries.

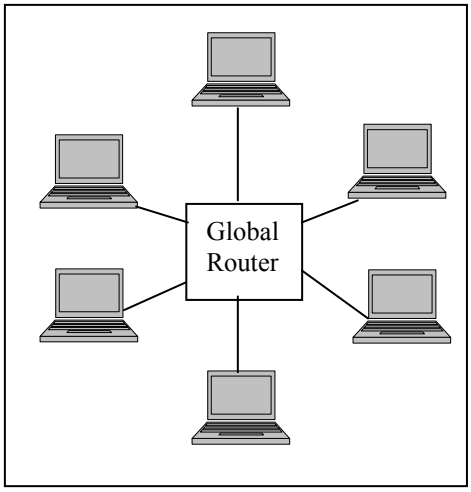
The implementation of the bulletin board is configured, in the code, for use by particular agents. Thus for example, agents in a particular region will only read the current prices posted for commodities in their particular region. Similarly, agents selling a commodity in more than one region will post region-specific prices for each region. Thus, a firm in region 1 will not see prices posted for commodities in region 2.

Structure of the Communications Network for Financial Transactions

A new communications network has been developed for CommAspen. No previous Aspen model contained this type of structure. The purpose of the communications network is to represent more realistically the process by which agents communicate with each other across electronic media. The traditional message-delivery system, used in CommAspen and in previous Aspen models, features instantaneous receipt of a message once that message is sent. With the new network, delays are built into the process. Importantly, the new communications network does not supplant the traditional message-delivery system. Instead, it is used in tandem with that system to currently handle only financial transactions that are used for the sample telecommunications and banking problem.

Composition of the Network

The central hub of the communications network is a global router, a special agent in CommAspen. The router is connected to CommTerminals (communications terminals) that belong to the individual CommAgents in the simulation, forming what is known as a star topology (see inset to right).



Agents use their CommTerminals to send messages to other agents. In the current version of CommAspen, each message is sent in a single packet. A packet sent from one agent to another does not go directly to the other agent. Instead, the packet is transmitted through the global router. The router accepts as many packets as it has room for in its buffer and drops the rest. There is a prioritization scheme; basically, higher priority messages get “first dibs” on an empty buffer slot, but do not bump a lower-priority message from a filled slot. The packets sit in the router’s buffer for a small randomized amount of time and are ultimately forwarded to their final destination.

The only characteristic in this entire communications system that the user can currently affect through the input file is the size of the buffer in the global router. Other characteristics can be changed to simulate outages, but this is currently done through the

Router code by CommAspen programmers. This capability will be added to the input file variables in future versions of the model. The buffer-size input parameter is called BankRouterBUFSIZE, which is located in the Run Data portion of the input file. With one router forwarding packets and the user having some measure of control over the number of packets the router can handle at one time, we can analyze the router's behavior in detail to ensure it is behaving correctly. *Importantly, the CommTerminals are also routers, but with some additional capabilities beyond that found in the global router*.

Message Passing Across the Network

Figure 8 illustrates in general how a message pertaining to a bank transaction is passed between two CommAgents, A and B, through the global router. The actual transmission process in the communications network is a bit more complex than described in the previous paragraph, as we will explain momentarily.

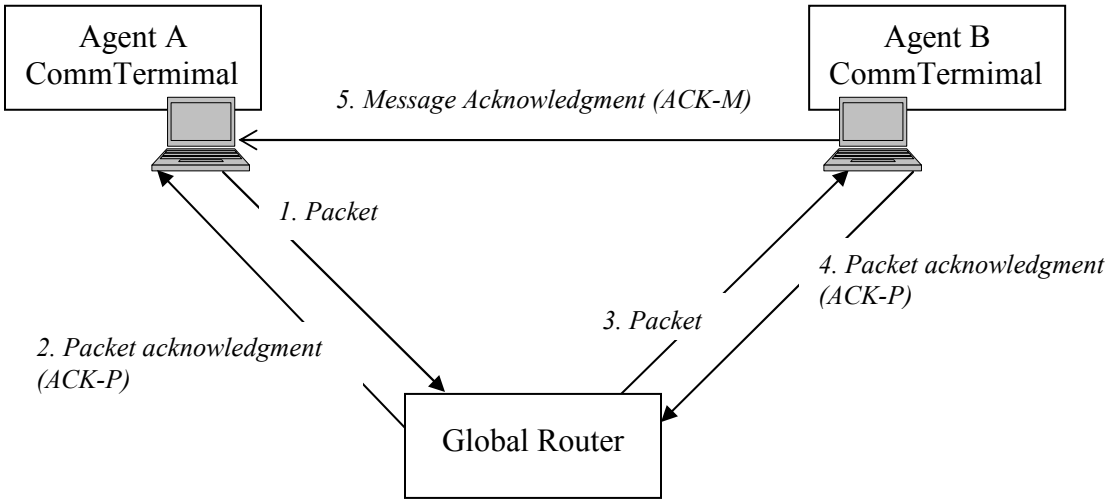


Figure 8. Message passing on the communications network.

The goal of the communications network is to simulate real-world delay in communications messages (like "clear bank check") from the sender to the receiver. For that reason and to introduce network protocols into the system, the path of such a message from sender to receiver incorporates not only delays in the global router but communications back and forth between the CommAgents and their respective CommTerminals and communications back and forth between the CommTerminals and the global router. In fact, the actual message contents regarding the financial transaction is not even sent across the communications network. The packets only contain information for the acknowledgments that occur at various steps in the process.

There are two levels of acknowledgments in the system: packet acknowledgment (ACK-P) and message acknowledgment (ACK-M). ACK-P messages roughly correspond to the hardware or wire-level protocols that move real packets between real routers, and to about half of the Transmission Control Protocol (TCP) or application-level protocol

that acknowledges the arrival of packets. The rest of the application-level protocol and the model's need to know when the original communication message, e.g., "clear bank check," can be scheduled for delivery are handled by ACK-M messages.

For all practical purposes, the packets are representational. Effectively, the sender is prevented from sending the message until the network delivers the message and is told to release the message, i.e., as a result of step 5. Thus, the CommTerminal of the sending agent actually holds on to the real message until it is told to release it by the receiving agent. When the CommTerminal finally "releases" the held message, it uses the pre-existing instant-delivery message system to schedule the message on the CommAspen calendar for immediate delivery.

Here are further details about the five steps identified in Figure 8.

1. The CommTerminal for Agent A has accepted a communication message from Agent A. The CommTerminal has logically divided the message into packets; for simplicity, we assume that the entire message fits into one packet. The CommTerminal then switches to "router mode," puts the packet in its router buffer, and sends the packet to the global router.

2. The global router accepts the packet, putting it into its buffer, and sends a packet acknowledgement (ACK-P). Agent A's CommTerminal, acting in router mode, frees up the buffer slot formerly occupied by the packet.

3. The global router examines the packet and sees that it is headed to Agent B. Following a set of its own internal processing rules, which include examining message priorities and introducing some amount of delay, the global router sends the packet to the CommTerminal (in router mode) for Agent B.

4. The CommTerminal for Agent B, acting in router mode, receives the packet and sends an ACK-P. The global router receives the ACK-P and frees up the buffer slot that the packet occupied during its stay. The CommTerminal for Agent B, still acting in router mode, sees that the arriving packet was intended for the CommTerminal, so it switches from router mode to CommTerminal mode to handle the packet.

5. The CommTerminal of Agent B, acting in CommTerminal mode, sees that the arriving packet has sequence number 1 of 1, and so it knows the "whole communication message" has arrived. Since it knows the entire message has arrived, Agent B's CommTerminal is *supposed* to have the complete message in hand ready to be given to Agent B. But the CommTerminal of Agent A actually has the message, waiting to release it. So the CommTerminal of Agent B sends the ACK-M message directly to the CommTerminal of Agent A to immediately release the message. (i.e., the ACK-M is not sent back through the network because doing so would cause further delay in an *already received* communication.)

The Router and Its Outage Implications

The presence of the router in the communications network is one method for modeling a communications congestion in CommAspen. Delays in the communications network are presently caused by heavy use—too many agents trying to send messages simultaneously, which fills up the router's buffer so that it stops accepting any more messages.

To simulate a router outage for the case studies presented below, CommAspen programmers added a length of time to the normal router delay. This had the effect of stopping the router from sending any messages for the specified period of time. In future versions of the model, the user will be able to simulate an outage through the input file.

The Problem

To test the new communications network in CommAspen, we have chosen to model interactivity and interdependency between the telecommunications and banking infrastructures. This section introduces characteristics of the problem we are studying. First, we briefly discuss the importance of telecommunications to banking. Then we identify the agent types that were constructed for the problem, followed by a look at their buying and selling interrelationships. Finally, we explain how the financial transactions are modeled in the new communications network for CommAspen.

Telecommunications and Banking Interdependencies

Using local and long-distance lines and services, financial institutions depend on telecommunications companies for the transmission of account information among themselves and the Federal Reserve [8]. This interdependence between the telecommunications and banking infrastructures, as well as many other critical infrastructures, became painfully clear during the tragic events of September 11, 2001. As John S. Tritak [9], director of the Critical Infrastructure Assurance Office, Bureau of Industry and Security, U. S. Department of Commerce, stated in a hearing before the U. S. Congress in July 2002: "That the loss of telecommunications services can impede financial service transactions and delivery of electric power is no longer an exercise scenario. There can be no e-commerce without e-electricity. There can be no e-commerce without e-communications."

The widespread destruction of supporting physical infrastructure for financial institutions near the World Trade Center, as well as extensive telecommunications breakdowns throughout the region during the week of September 11, caused dislocations in financial markets. For four days, U. S. equity markets closed. For two days, most bond trading halted, resulting in significant disruptions of clearing and settlement mechanisms for government securities, commercial paper, and repurchase agreements [10]. According to the U.S. Department of Homeland Security [2], the closure of the equity markets did

not occur because the markets or market systems were inoperable. Rather, the markets closed because the telecommunications lines in lower Manhattan that connected major market participants were so damaged that they could not be restored immediately.

The risk environment for financial institutions was significantly heightened by the events of September 11. Lessons-learned analyses have been conducted by government and industry that focus on developing more robust business-continuity plans across the financial industry [2,10,11]. Of critical importance is the need for financial institutions to build redundancy and backup into their systems and operations. One of the major vulnerabilities identified post-9/11 was concentration of a financial institution's primary and backup sites in a small geographic area. Without telecommunications, a primary site had no access to its backup site, even though in some cases financial institutions had contracted with multiple telecommunications providers [10].

In April 2003, the Federal Reserve, the Office of Comptroller of the Currency, and the Security and Exchange Commission [11] published an interagency paper that included a set of business-continuity objectives for financial firms. These objectives are as follows:

- "Rapid recovery and timely resumption of critical operations following a wide-scale disruption;

- Rapid recovery and timely resumption of critical operations following the loss or inaccessibility of staff in at least one major operating location; and

- A high level of confidence, through ongoing use or robust testing, that critical and internal and external continuity arrangements are effective and compatible" (p. 5–6).

Regarding the second objective above, a number of firms had expressed concerns to the government agencies about the resilience of the telecommunications infrastructure, which would affect their ability to recover. The authors responded that by establishing geographically dispersed facilities, financial firms can achieve additional diversity in services by telecommunications and other infrastructures.

Efforts to bring together financial and telecommunications services are under way. BITS, the technical arm of the Financial Services Roundtable, a banking industry trade group, has been working to develop a closer relationship with telecommunications carriers and the federal government [8]. As part of this effort, the executive director of BITS attended a meeting sometime in 2003 in which a dissertation by a graduate student at George Mason University was presented and discussed. The student demonstrated a laptop database he had constructed (using the Internet) in which he had mapped every business and industrial sector in the U.S. economy, overlaying these with the fiber-optic network that connects them. By clicking on a bank, say, in Manhattan, one could see who has communication lines running into the facility and where these lines are located. Meeting attendees, as well as government officials, were all concerned about the sensitivity and the eventual publication of these data. Many attendees were not aware how

interdependent their banking systems are with telecommunications. Plans were made after the meeting for the financial industry, the telecommunications industry, and the National Communication System to jointly simulate a bomb attack and cyber assault to measure the impact on financial services. The simulation was to be held in an undisclosed Midwestern city in the summer of 2003 [12].

Agent Types in the Problem

For our problem, we have constructed several types of agents, with the following user-defined names:

- Consumer
- Firm_A, Firm_B, Firm_C, Firm_D, Firm_E, and Firm_F
- Bank

The Consumer type of agent is configured to consume certain commodities and to buy such commodities from several types of firm agents. The Consumer has a bank account and can write checks for its purchases. The Consumer is considered to be the ultimate sink for commodities in the problem.

The six types of firm agents are configured to buy and sell commodities among themselves and to consumers as well. Structurally, all firm agents use the FirmProduction block and the FirmBuyer blocks, which allow us to create unique types of agents quickly. Each type of firm agent has a bank account and can write checks for its purchases.

The Bank is a special type of agent that is responsible for clearing checks from other agent types and reconciling accounts with its account holders, i.e., all those other agent types that have accounts in a particular bank. There are few user inputs for this type of agent.

Commodity Exchange among Agents

There is a web of complex interrelationships between the various buyer and seller components of the agent types constructed for the problem. Figure 9 illustrates the buyers and sellers of particular commodities. Those agent types that buy commodities from more than one other agent type have either a production recipe that they use or a list of commodities that they consume. In the problem, each firm type has an associated production commodity of a similar name, e.g., Firm_A has output called COMPONENT_A; the output of Firm_B is COMPONENT_B.

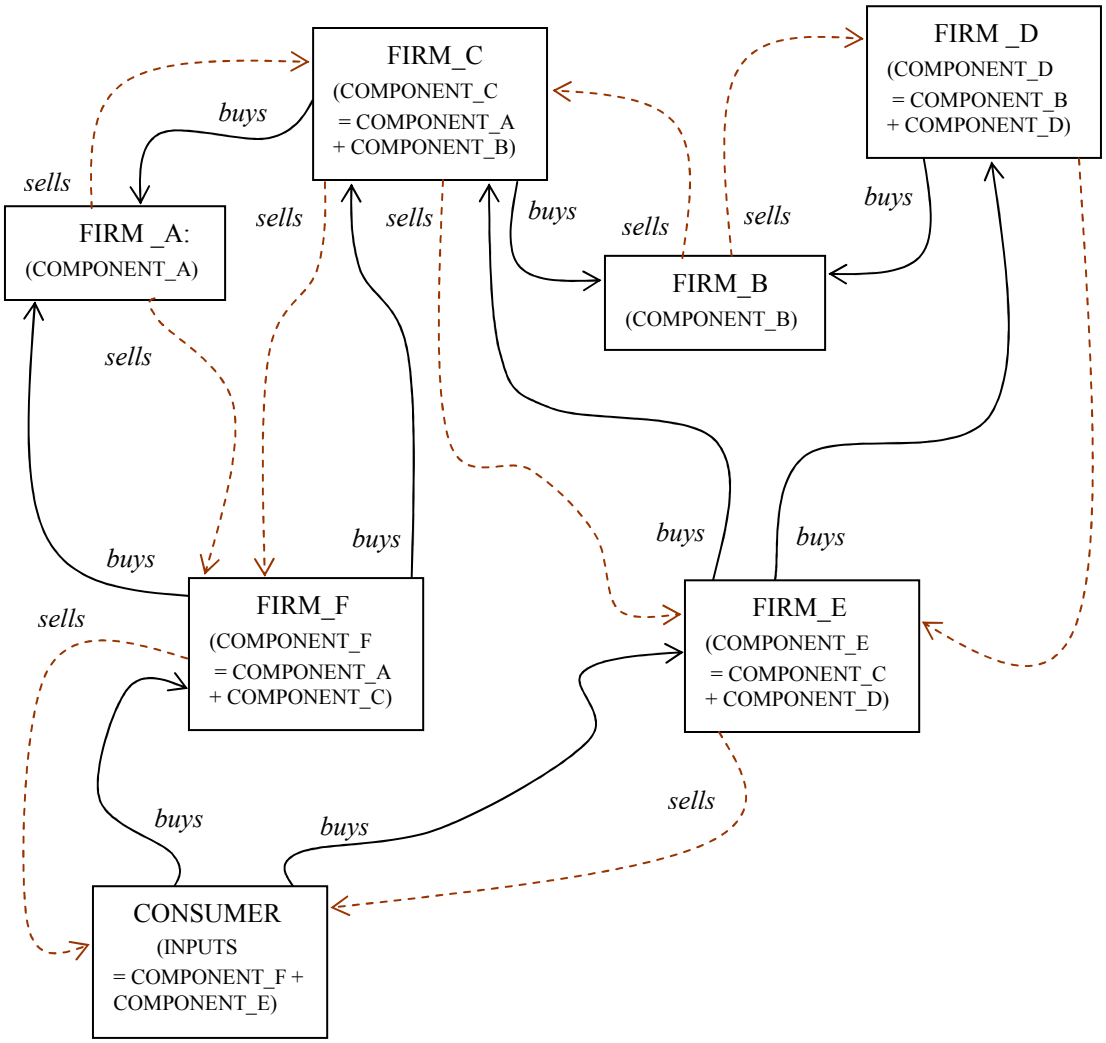


Figure 9. Commodity buyers and sellers in the problem.

As an example of the above exchange relationships, FIRM_A sells one commodity, COMPONENT_A, to both FIRM_C and FIRM_F. Both FIRM_C and FIRM_F need COMPONENT_A as part of their recipes to produce their output, COMPONENT_C and COMPONENT_F, respectively.

To purchase commodities, the individual consumer and firm agents write checks. These checks are transmitted to the selling firms via messages over the communications network and then to the bank for clearance, as explained in the next subsection. We are particularly interested in what happens when the communications network, which is exclusively used here for financial transactions, slows down. Do firms run into cash-flow problems because they are not being paid for sold goods? How does that affect the economy?

In our problem, we have not explicitly created an agent to represent the telecommunications infrastructure. Instead, the router in the communications network is representative of the telecommunications infrastructure.

How Financial Transactions Are Modeled

CommAspen has a special check-processing system that enables agents to purchase commodities from agents who sell those commodities and pay for the commodities by check. Bank agents are responsible for clearing and reconciling the checks. Note that no agent can purchase anything unless it has sufficient assets, which must be in the form of "ready cash." That function is handled by the Accountant that is built into the basic CommAgent functionality. Thus, all checks that are written should clear.

Figure 10 depicts a very simplified representation of the check-clearing process. For example, acknowledgements between CommTerminals and between CommTerminals and the router, as discussed in the previous section, are not shown. In the figure, we use a consumer agent, a firm agent, and two bank agents. The check-clearing process involves the pathways a check takes from the time it is sent by the purchasing agent until the time the check is actually reconciled by the bank that holds the purchaser's account. Although for the sample run we have used only one bank, this example illustrates the current capabilities of the model in the check-clearing process to handle interbank transfers. As the simulation runs for our problem, however, this process is much more complex, because many agents are sending checks that must all go through the router at various points. Note that banks maintain an accounting system of their own customers.

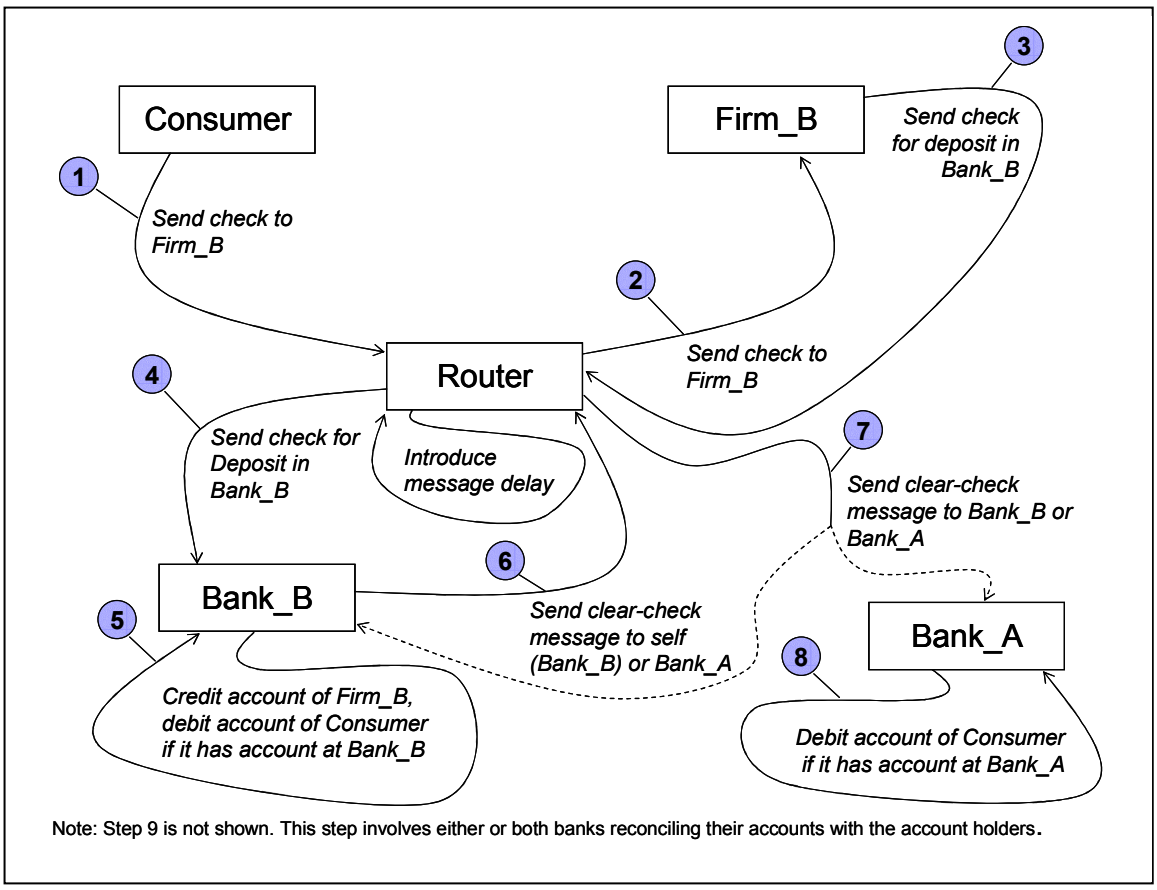


Figure 10. Example of the check-clearing process.

The sequence below provides further details for the various steps in the process. The term *router* in these steps refers to the global router.

1. One agent (in this case, Consumer) sends a check to another agent (here, it is Firm_B). The check writer could have an account in the same bank as the agent to whom the check is written or an account in a different bank; hence, the presence of Bank_A, which in the example could belong to the consumer. The check must go through the router.
2. After handling the check according to its message-processing rules, the router sends the check to Firm_B.
3. Firm B sends the check to Bank_B, its own bank. The check must go through the router.
4. After handling the check according to its message-processing rules, the router sends the check to Bank_B.
5. Bank_B credits Firm_B's account. If the consumer has an account at Bank_B, Bank_B will debit the consumer's account.

6. Bank_B sends a clear-check message to itself if the consumer's account is at Bank_B. If the consumer's account is at Bank_A, Bank_B sends the clear-check message to Bank_A. The clear-check message must go through the router.
7. After handling the clear-check message according to its message-processing rules, the router sends the message either to Bank_B or Bank_A.
8. If the clear-check message is received by Bank_A, that bank debits the account of the consumer.
9. At some later point, Bank_B reconciles the account with Firm_B and with the consumer if it also has an account with Bank_B. Otherwise, Bank_A will reconcile the consumer's account. The reconciliation activities are also done through messaging, but the messages are not passed through the router.

The delays introduced by the router after steps 1 and 3 prevent Firm B from being able to spend the funds. The delay in the router as a result of step 6 has little effect, but it adds traffic to the router. By introducing a delay for step 9, the process prevents both the consumer and Firm_B from knowing whether any deposited checks (from step 5) have arrived, i.e., effectively cleared, and that the associated funds are ready to be spent.

The approach taken in the check-clearing process enables banks to make the funds from all deposited checks (even for very large amounts) available immediately. This policy decision can be changed, but it adds more steps to the whole transaction. The funds-availability policy is really independent of the underlying network structure.

Note: *The behavior described in step 6 was changed as this document was going to press. In this new behavior, any message the agent sends to itself will not go to the "global" router. Instead, the message gets chopped up into packets, and the packets are passed to the router part of the CommTerminal (as identified in* ***Message Passing Across the Network****). Then, the router part sees that the packets are destined for itself and passes them back to the CommTerminal. Figure 10 does not reflect this behavioral change.*

Results and Analysis

This section presents the initial run conditions, discusses additional characteristics of the test problem, and provides an analysis of the results.

Run Conditions

Simulations using the model were run with the following individual agents:

- 100 Consumer agents
- 10 Firm_A agents
- 10 Firm_B agents

- 7 Firm_C agents
- 4 Firm_D agents
- 3 Firm_E agents
- 3 Firm_F agents
- 1 Bank agent

Test Problem

One basic test problem was used in all of the analysis runs. The problem is a supply chain with Consumer agents driving demand for goods at the end of the chain. The problem is graphically depicted in

Figure **11**. The integer values on the lines that connect the agents represent the input requirements for making the downstream commodity or the demand requirements in the case of Consumer agents. For example, type F firms require two units of A and two units of C to make 1 unit of F. There was also one Bank agent in the problem that is not depicted in the graph. Various individual parameters such as production level and router buffer size were altered for each test run. These parameters will be described in each of the analysis sections.

Error!

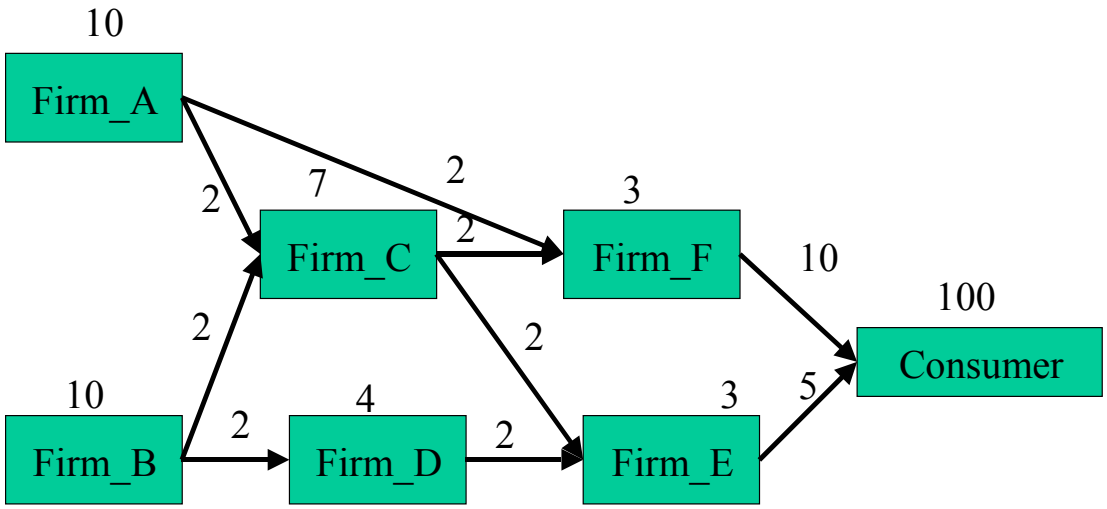


Figure 11. Supply chain for test problem.

Analysis of the Router Buffer Capacity

As a first analysis, we perturbed the size of the finite communications buffer and evaluated the impact on the percentage of dropped packets. An analysis was done for a

buffer size of 7, 9, 10 and 12. At a buffer size of 12, there were no packets dropped during the simulation.

We would expect the number of dropped packets to increase supralinearly as the buffer size is reduced, and that is observed in the test-case simulation. We define here the concept of severely errored time steps (SETs) by analogy to the term *severely errored seconds* used by the telecommunications industry. Severely errored seconds is a measure of the degradation of transmission lines [13]. Here, a SET is a measure of the degradation of a packet connection between agents, and is defined as the percentage of dropped packets in a time step that exceeds a particular threshold. Figure 12 presents a histogram of SETs for each of the three buffer sizes, as the threshold in the definition of SETs is varied. The increase in dropped packets when the buffer size is reduced from 9 to 7 is significantly greater than twice the increase in dropped packets when the buffer size is reduced from 10 to 9.

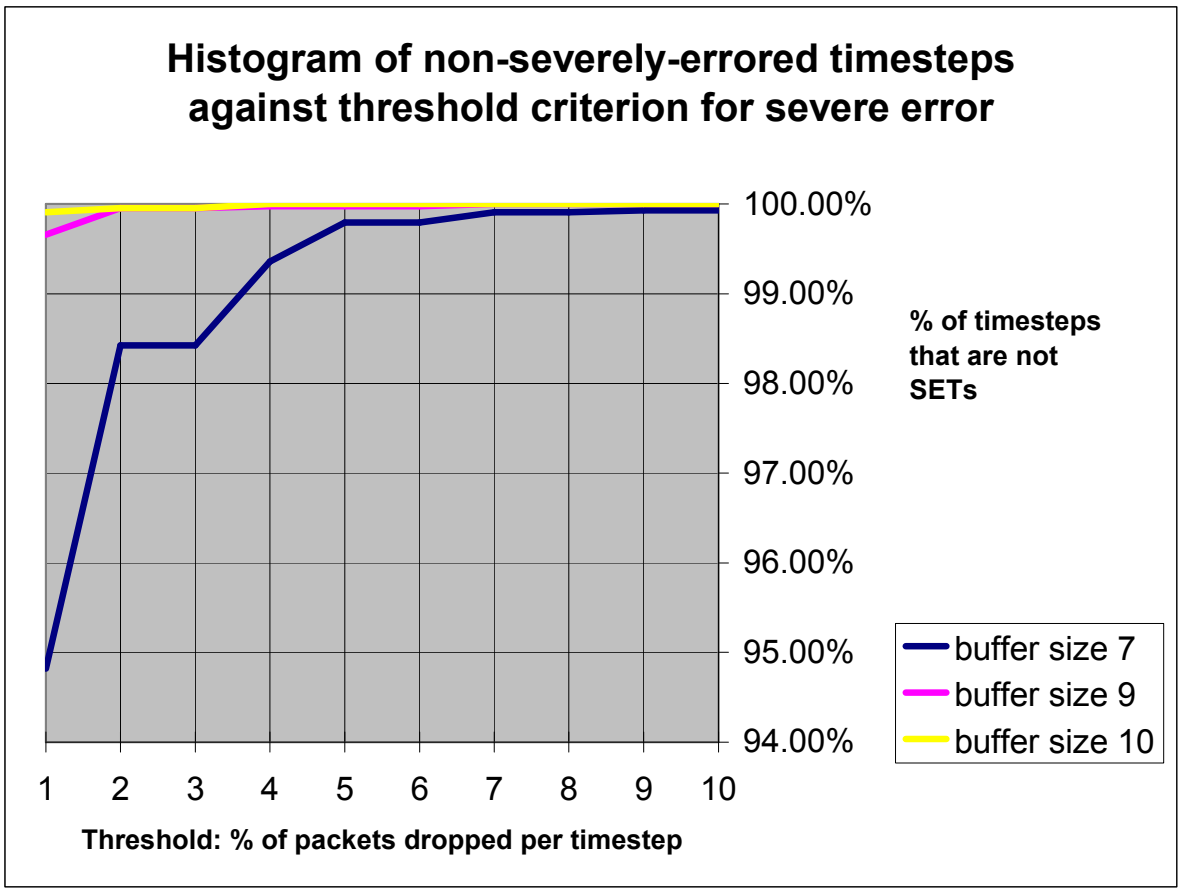


Figure 12. Histogram of SETs for the three buffer sizes.

For example, for a SET threshold of 10% packet drop, 99.93% of the time steps are not SETs for a buffer size of 7; whereas, no time steps are SETs for buffer sizes of 9 and 10. For a SET threshold of 1% packet drop, 5% of the time steps are SETs under a buffer

size of 7, 0.3% of the time steps are SETs under a buffer size of 9, and 0.1% of the time steps are SETs under a buffer size of 10. These results suggest that there are decreasing marginal benefits for increasing the size of the communications buffer, as would be expected.

Analysis of Router Outage

The router in this scenario is the communications router for the Bank agent. The router has a variable buffer size that was varied in a range of 7 to 14 packets. Once the buffer is full, the router no longer accepts new packets. These packets are dropped by the router. The agents that sent these packets (e.g., households, firms, etc.) may resend at another time or abandon their transmission.

The purpose of this analysis was to study the impact of an outage on the percentage of packets dropped by the router following resumption of communication services. The outage was tested as both one week and two weeks in duration. Aside from the length of the outage itself, there was not a significant difference in router performance following the outage as a function of the one- or two-week outage. There was more variation in performance as a function of buffer size. For buffer sizes greater than or equal to 10, there were no dropped packets prior to the outage. This was expected since the larger buffer size is sufficient to accommodate the communications traffic generated by all the agents used in the CommAspen test case. For buffer sizes of 9 or smaller, there are occasional dropped packets that increase as the buffer size is lowered to 7. For buffer sizes of less than 7, there are significant packet losses due to the small buffer size, which is not sufficient to accommodate the communications traffic in the test case.

As can be seen in Figure 13 through Figure 32, the impact of the outage is significant for a period of days and even weeks following the resumption of services. Traffic is not able to get through to its destination, and thus all packets are dropped during the outage. Following the outage, in every buffer size tested, there is a transition period in which the percentage of packets dropped is still high but declining as the router empties its buffer. This takes many time steps (days in fact) because agents resend messages that were dropped during the outage. This resending of messages continues to overload the router buffer though the backlog gradually decreases. At some point, the router buffer is emptied. This again is a function of the buffer size. The smaller the buffer size, the longer this takes. Once the router buffer is emptied, then in theory the communications network has completely recovered from the outage. However, it can be seen from Figure 23 through Figure 32 that there continue to be significant percentages of packets dropped following this emptying of the router buffer. This occurs despite the fact that in some of the buffer sizes, there were *no* packets dropped prior to the outage. This means that the agents themselves continue to have a post-outage effect that prevents the network from returning to its pre-outage state. It is postulated here (based on Figures 23 through Figure 32) that the communications terminals of the CommAspen agents have stored up messages they wish to send and have held them in reserve waiting for the router buffer to

clear. Thus even after the router buffer clears, the agents will continue to send messages that had been dropped during the outage or post-outage transition.

Packet Loss: No Outage

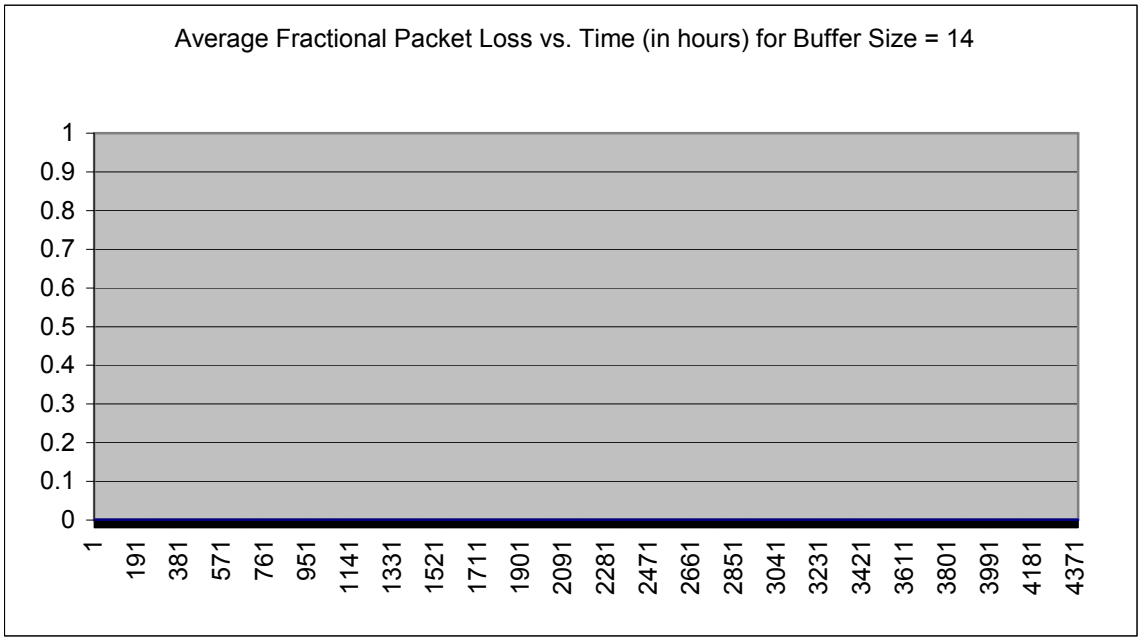


Figure 13. Plot of dropped packets versus time over a six-month period with no outage for buffer size of 14. In this case, the plot is zero for all time.

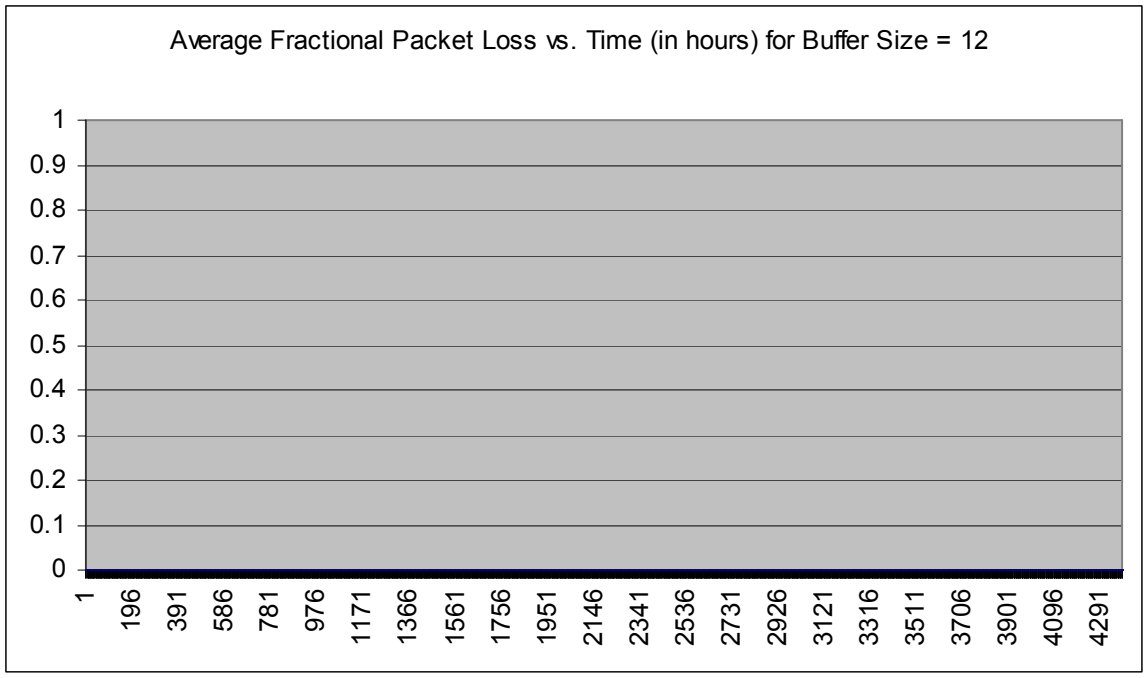


Figure 14. Plot of dropped packets versus time over a six-month period with no outage for buffer size of 12. In this case the plot is zero, for all time.

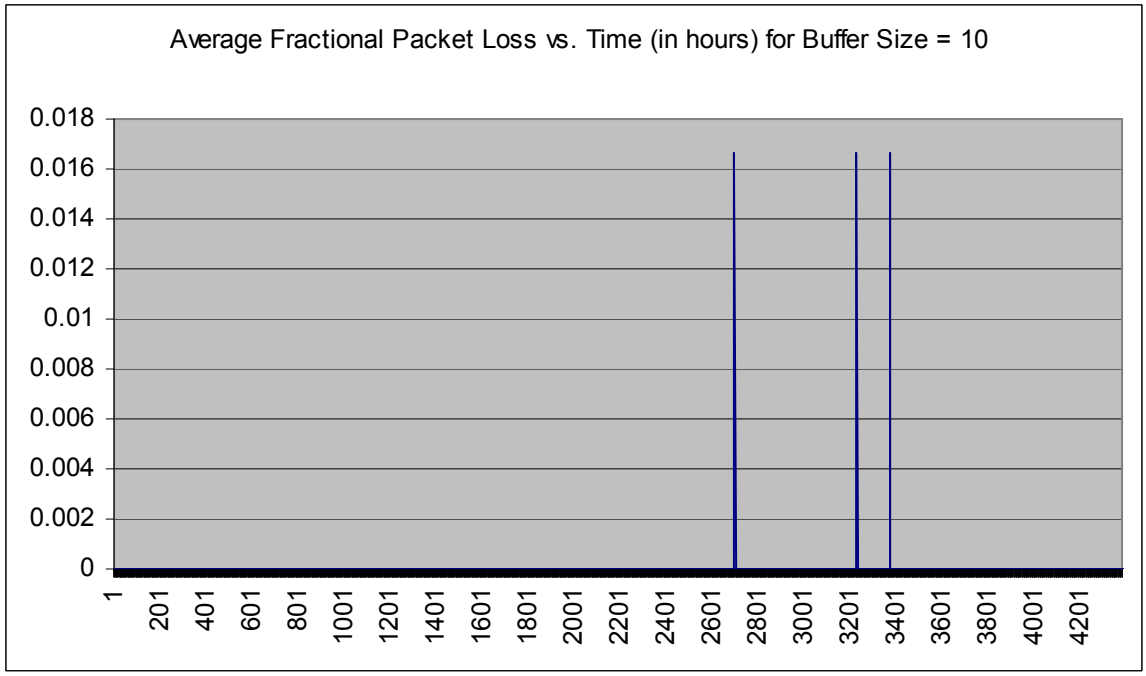


Figure 15. Plot of dropped packets versus time over a six-month period with no outage for buffer size of 10.

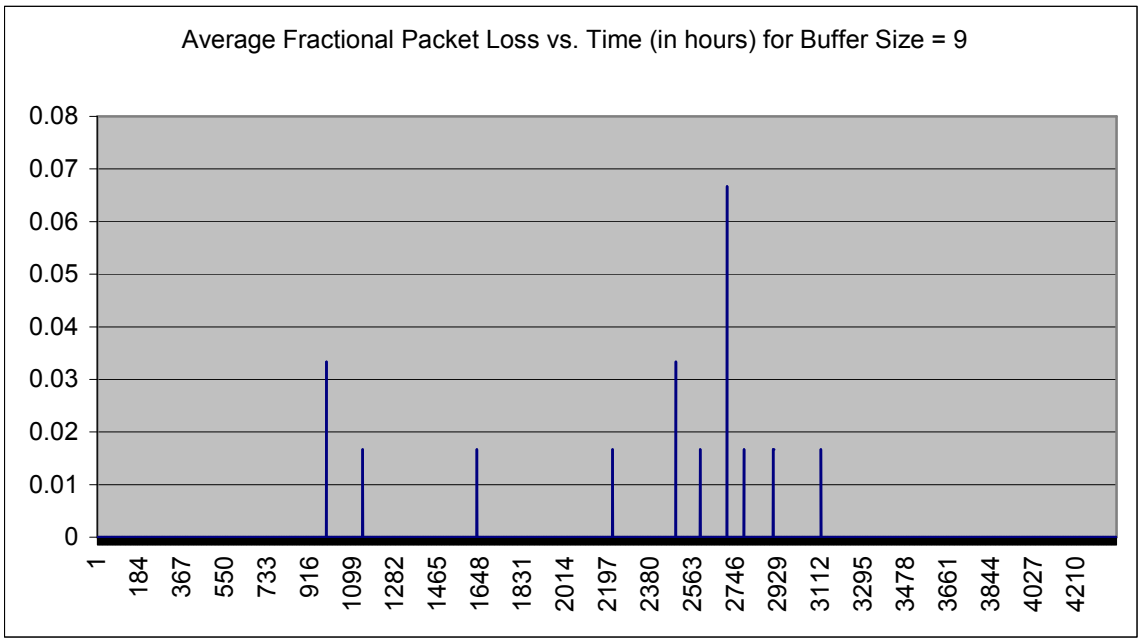


Figure 16. Plot of dropped packets versus time over a six-month period with no outage for buffer size of 9.

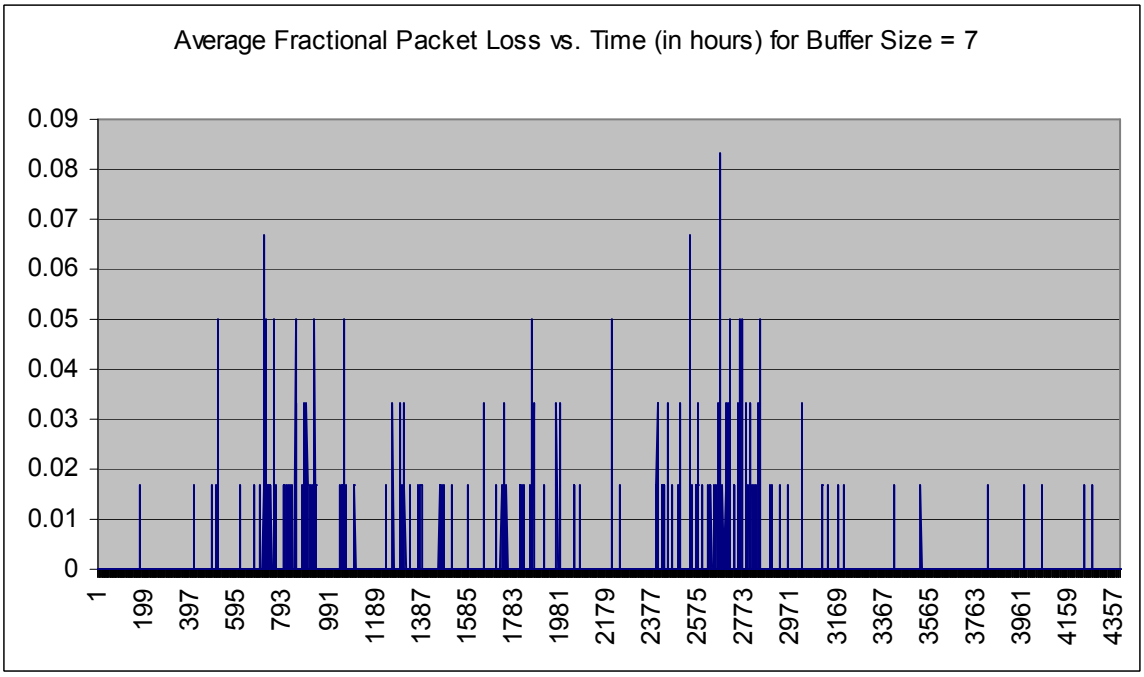


Figure 17. Plot of dropped packets versus time over a six-month period with no outage for buffer size of 7.

Buffer Fill: No Outage

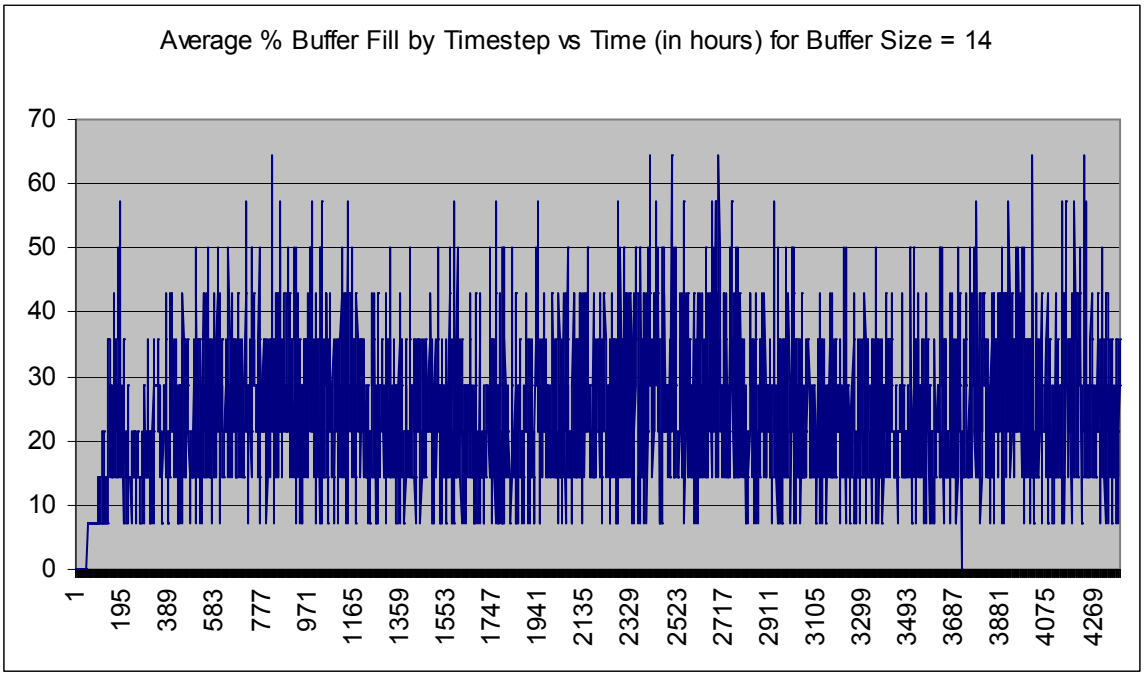


Figure 18. Plot of percent buffer-fill versus time over a six-month period with no outage for buffer size of 14.

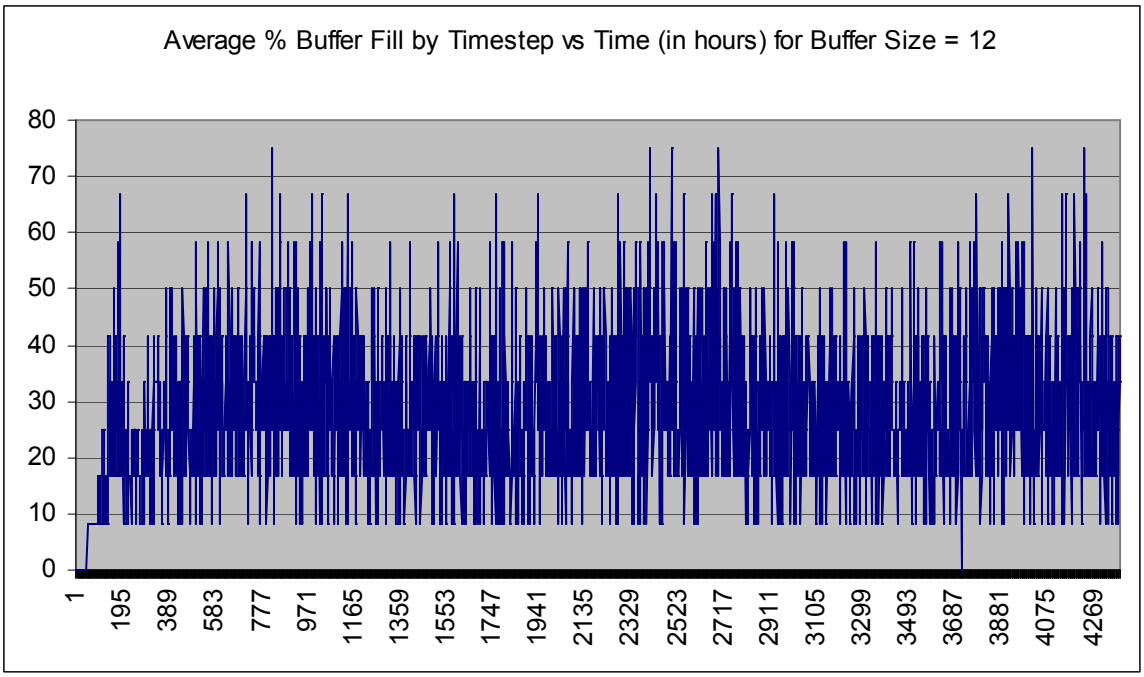


Figure 19. Plot of percent buffer-fill versus time over a six-month period with no outage for buffer size of 12.

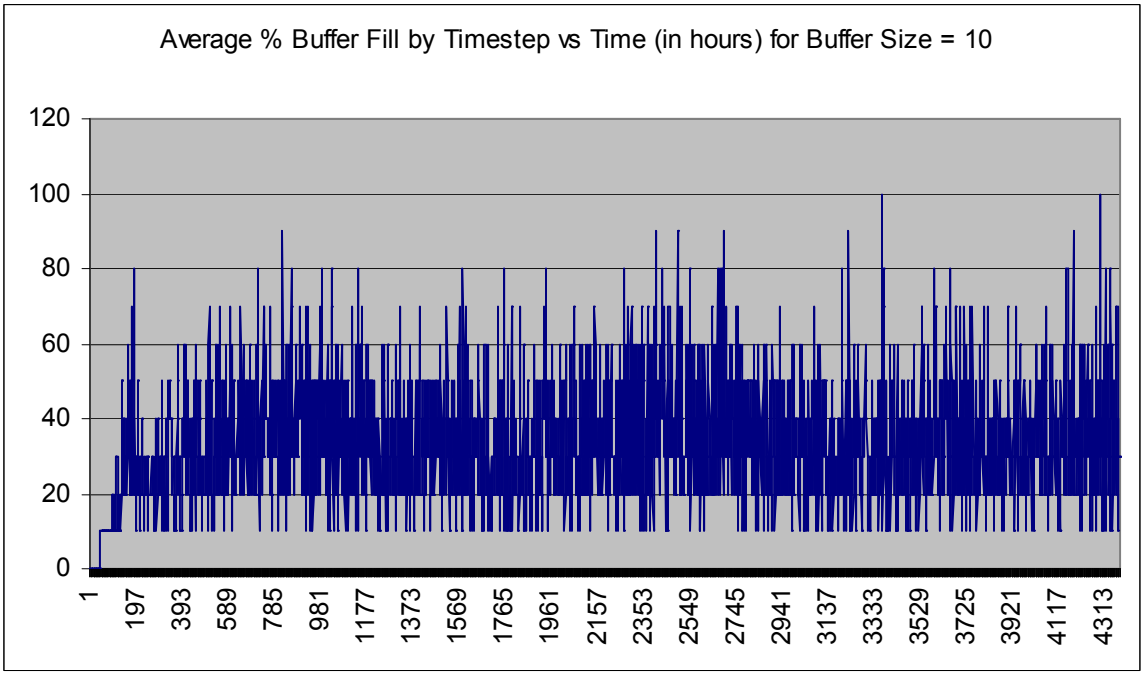


Figure 20. Plot of percent buffer-fill versus time over a six-month period with no outage for buffer size of 10.

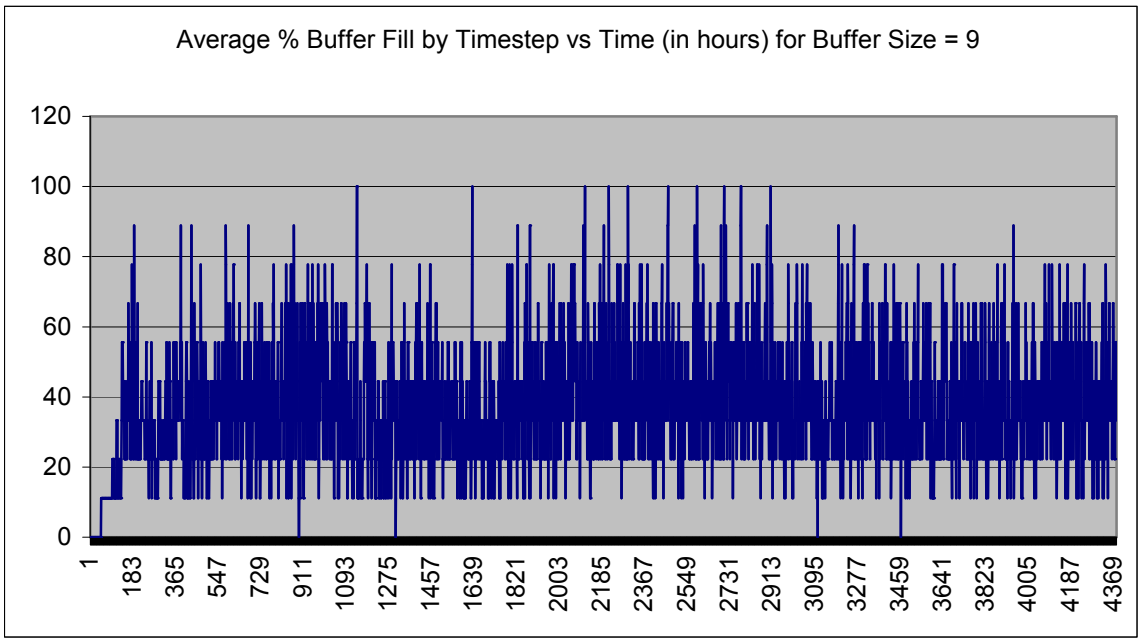


Figure 21. Plot of percent buffer-fill versus time over a six-month period with no outage for buffer size of 9.

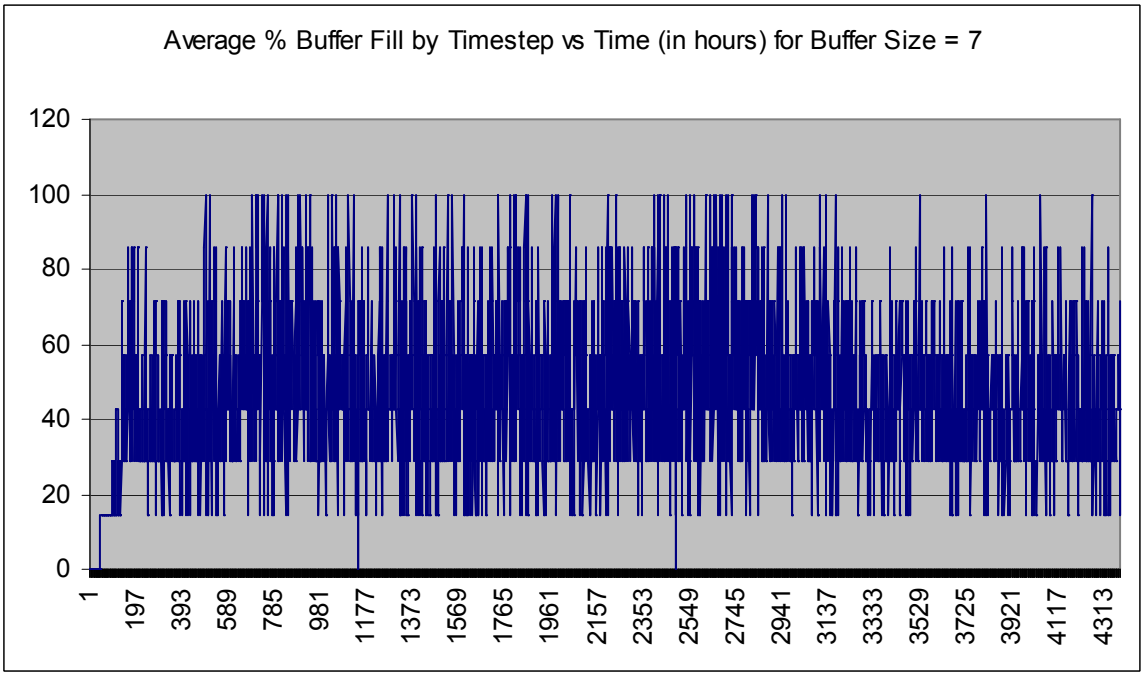


Figure 22. Plot of percent buffer-fill versus time over a six-month period with no outage for buffer size of 7.

Packet Loss: Two-Week Outage

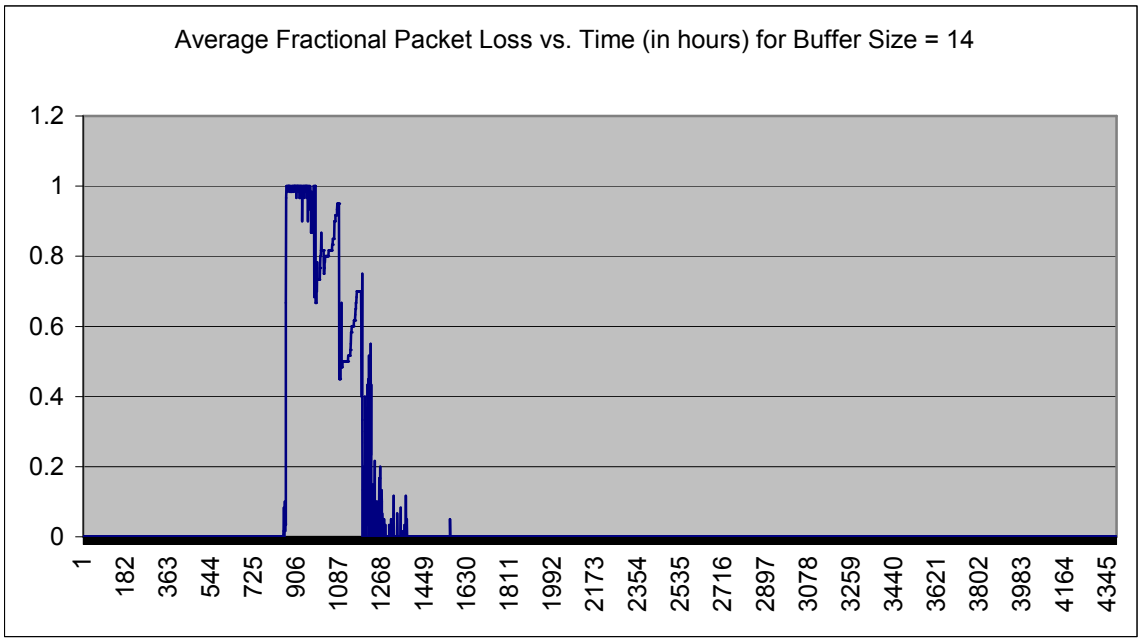


Figure 23. Plot of dropped packets versus time over a six-month period with a two-week outage for buffer size of 14.

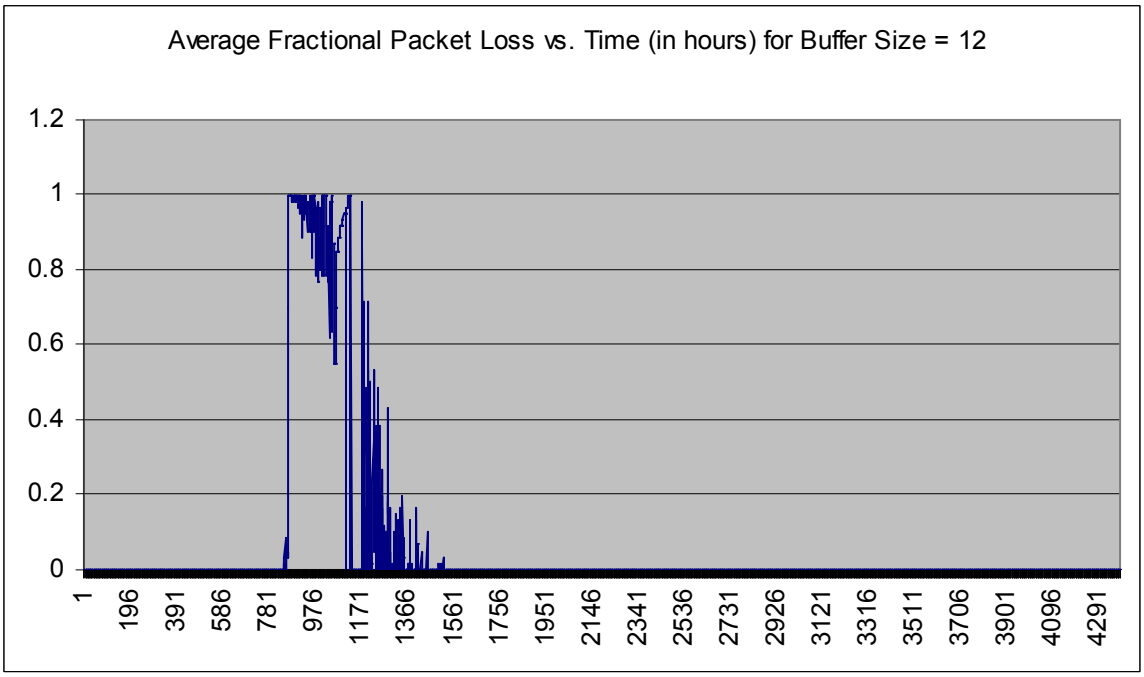


Figure 24. Plot of dropped packets versus time over a six-month period with a two-week outage for buffer size of 12.

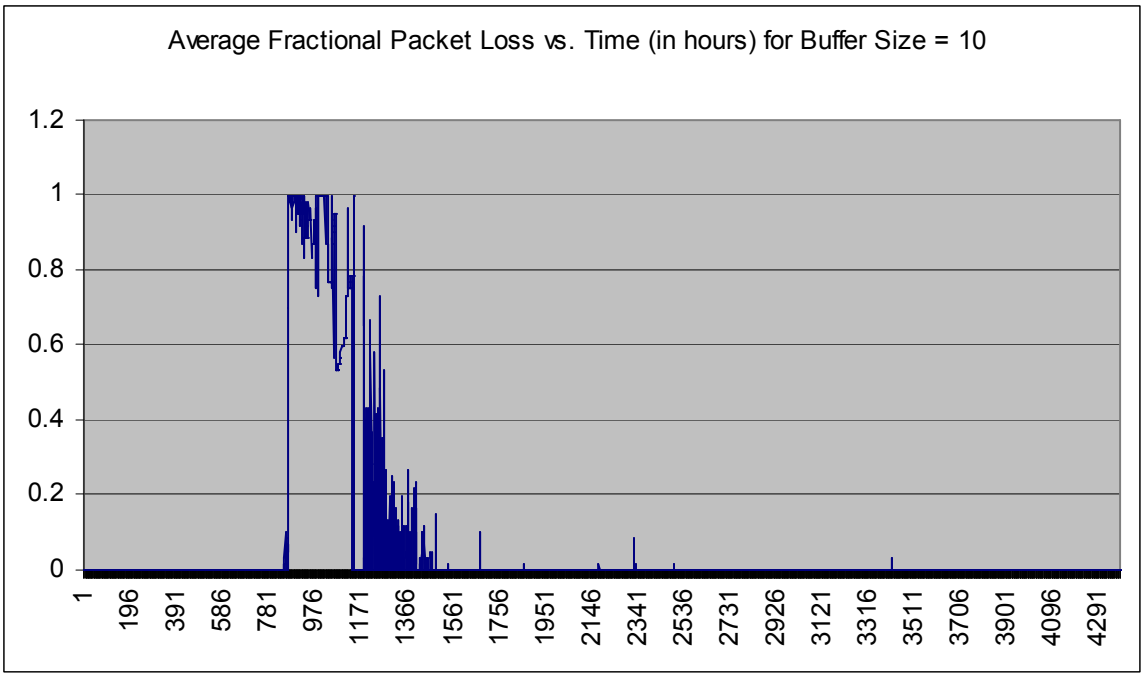


Figure 25. Plot of dropped packets versus time over a six-month period with a two-week outage for buffer size of 10.

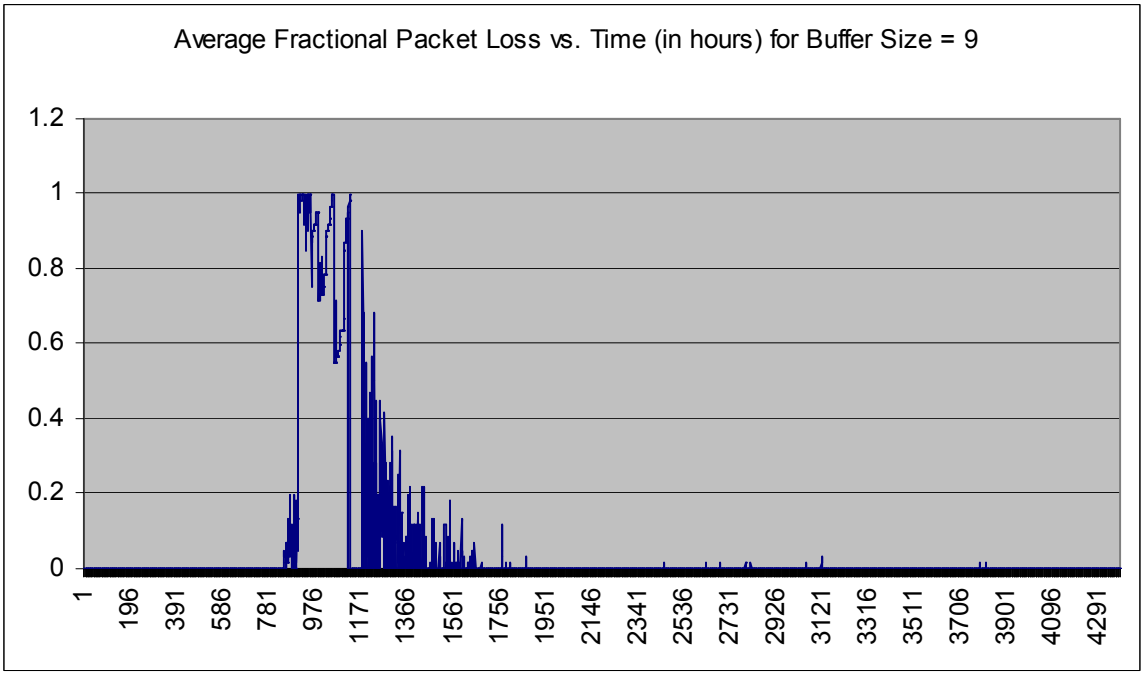


Figure 26. Plot of dropped packets versus time over a six-month period with a two-week outage for buffer size of 9.

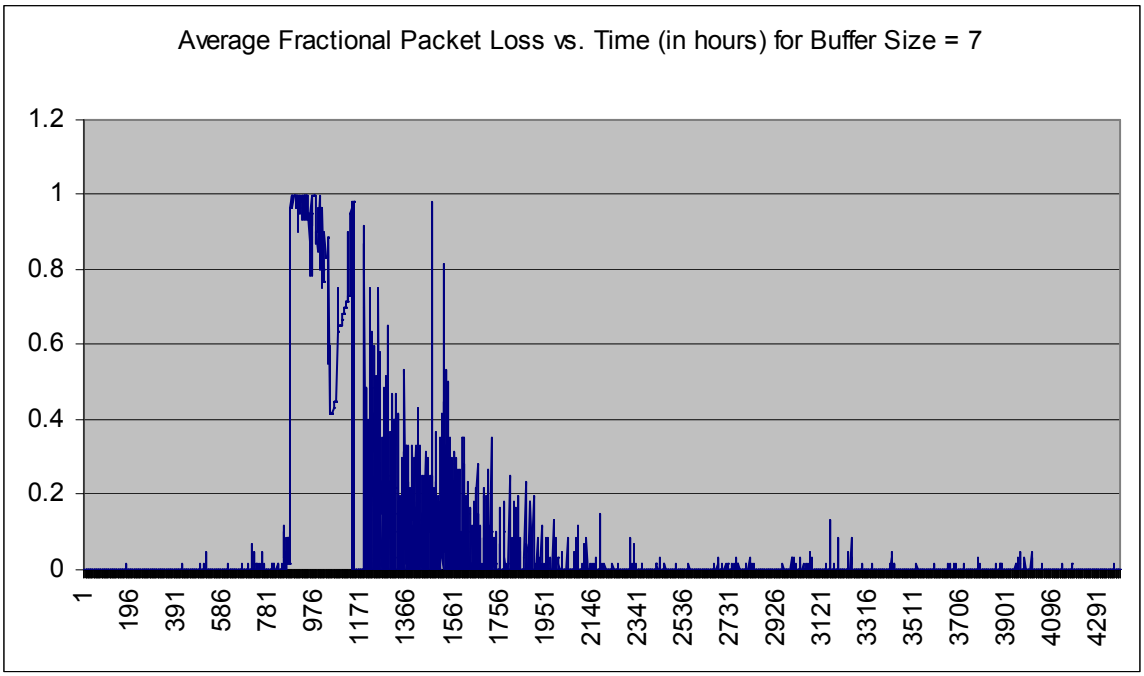


Figure 27. Plot of dropped packets versus time over a six-month period with a two-week outage for buffer size of 7.

Buffer Fill: Two-Week Outage

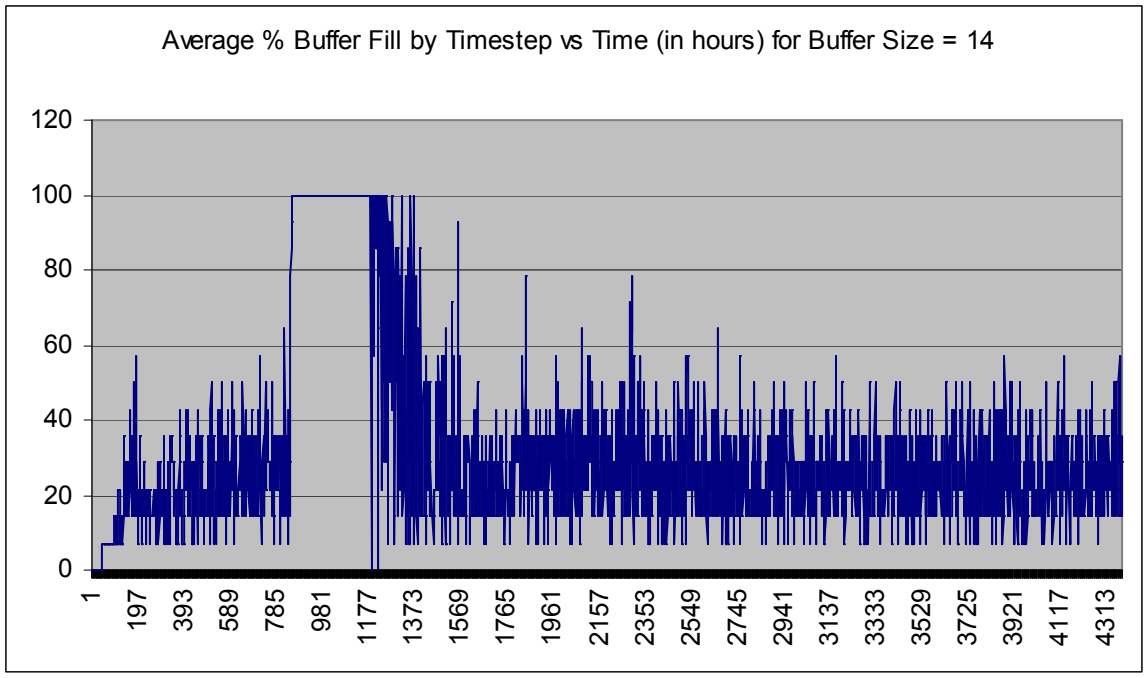


Figure 28. Plot of percent buffer-fill versus time over a six-month period with a two-week outage for buffer size of 14.

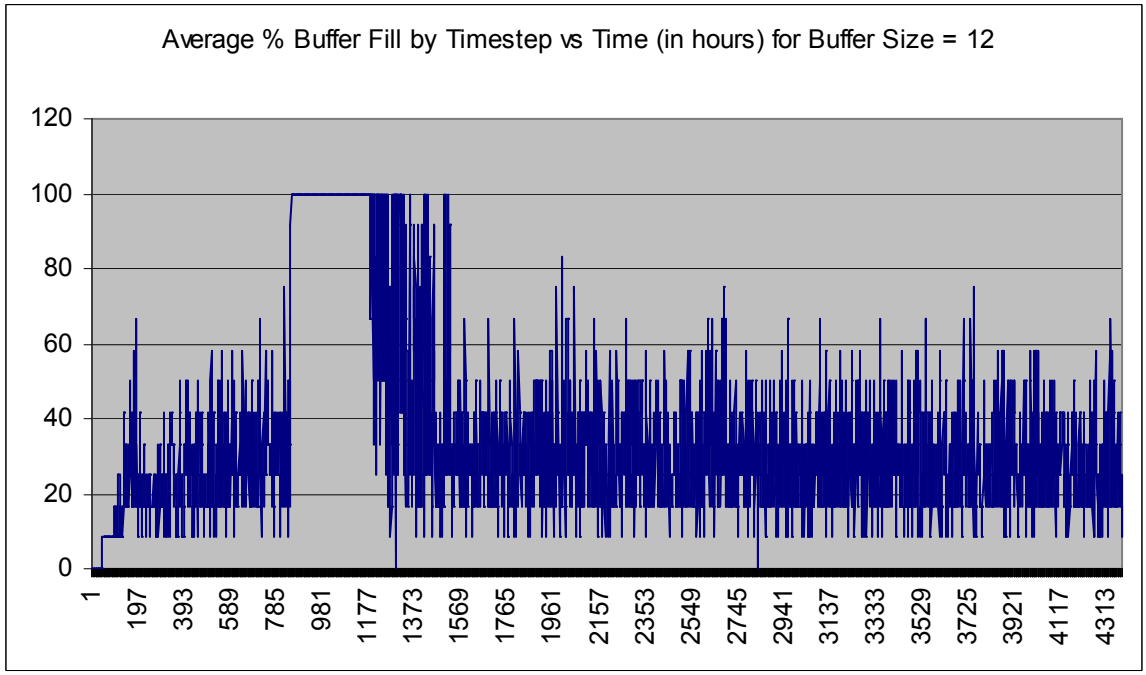


Figure 29. Plot of percent buffer-fill versus time over a six-month period with a two-week outage for buffer size of 12.

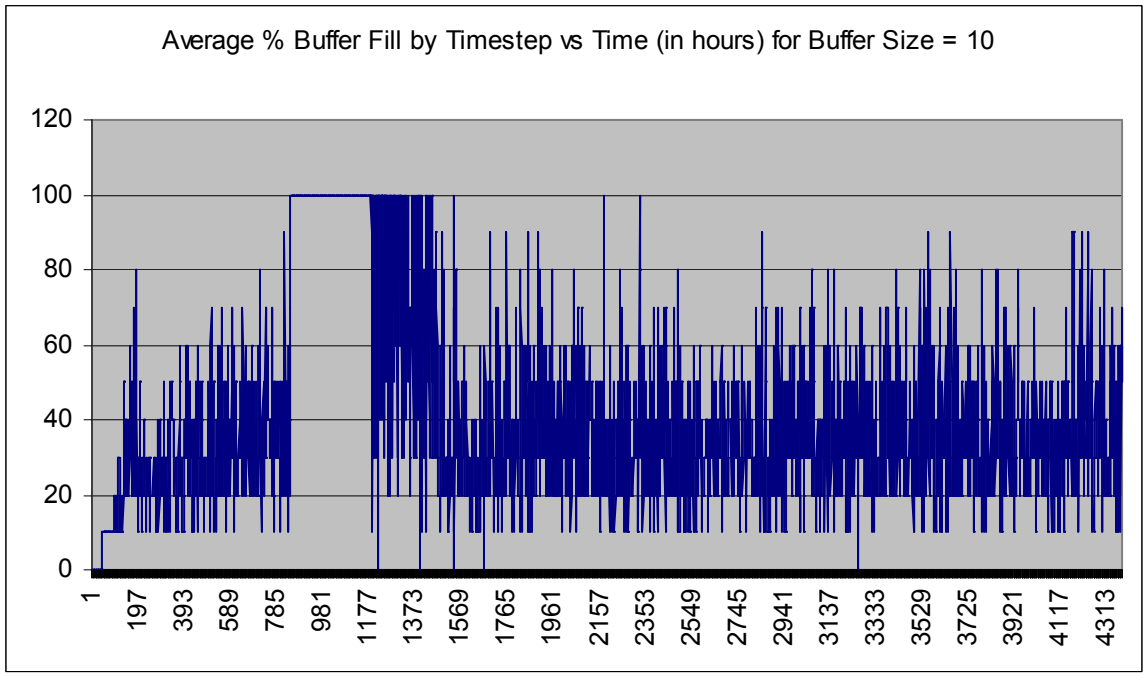


Figure 30. Plot of percent buffer-fill versus time over a six-month period with a two-week outage for buffer size of 10.

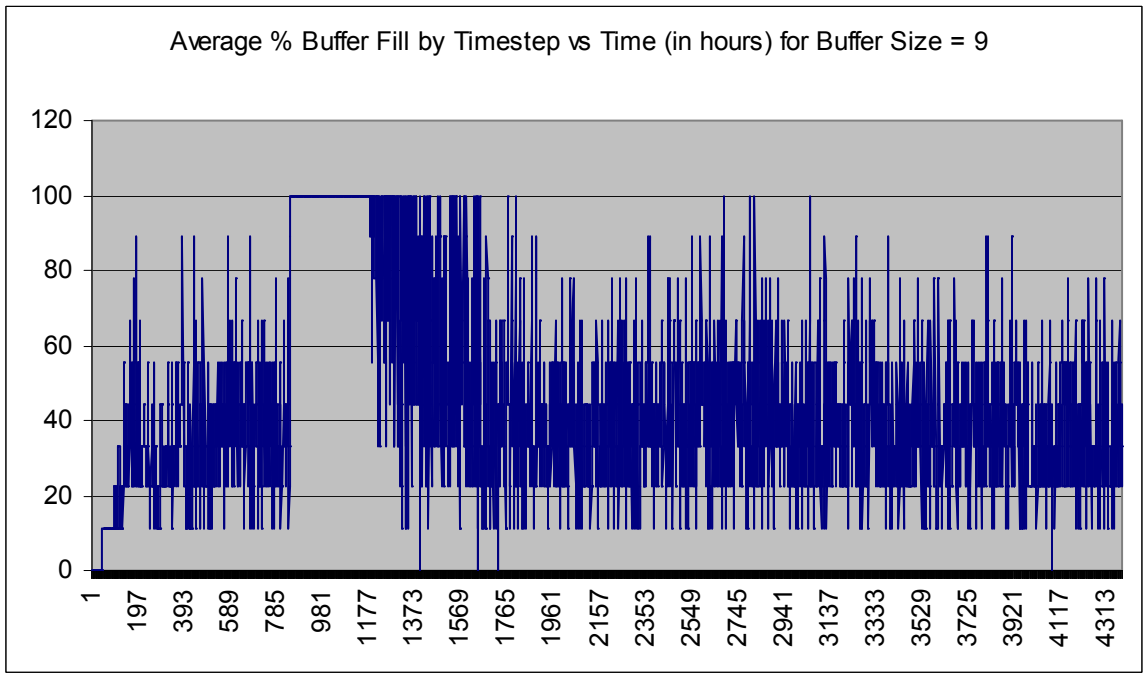


Figure 31. Plot of percent buffer-fill versus time over a six-month period with a two-week outage for buffer size of 9.

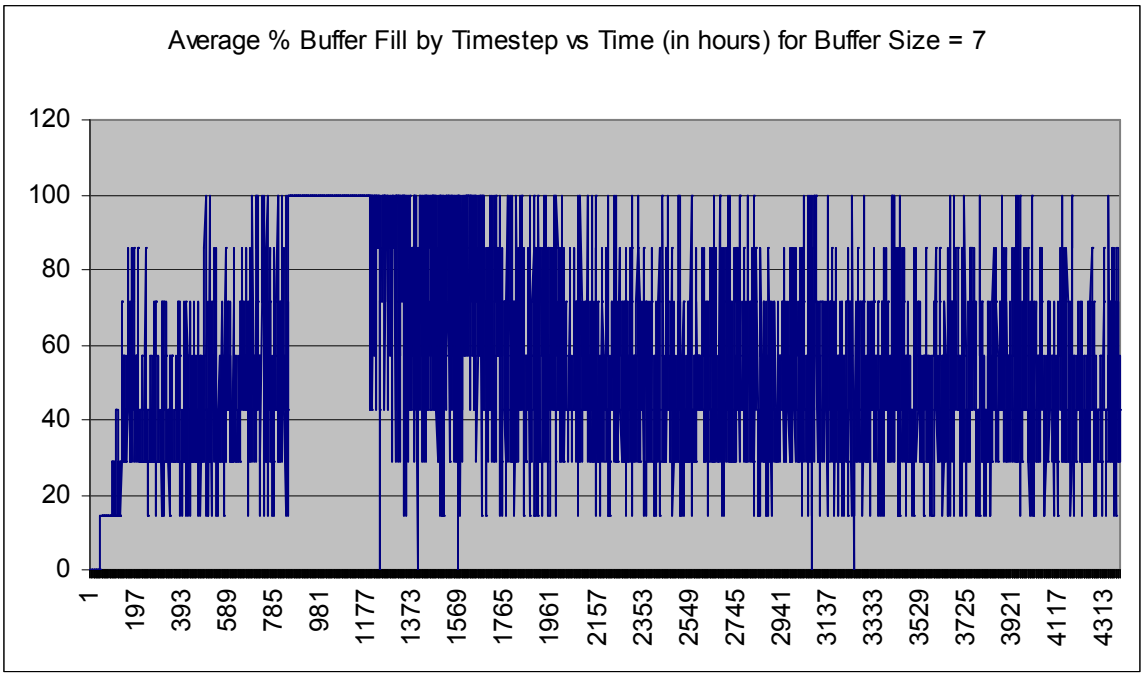


Figure 32. Plot of percent buffer-fill versus time over a six-month period with a two-week outage for buffer size of 7.

This behavior of resending messages implies a memory effect among the agents that exceeds the memory effect of the router itself. Thus, the continued nonzero packet losses following the clearing of the router buffer after an outage is explained by this theory. There may be means to handle this situation differently in the future. For instance, the agents could be put on an organized schedule following an outage. This schedule would dictate when agents can resubmit their dropped messages as well as their new messages for some period of time following an outage. This schedule could be determined by prioritizing the agents (perhaps certain businesses would have a higher priority than households, etc.) and also by the size of the router buffer. As the router buffer size is increased, this post-outage behavior is improved, but one could argue that traffic will always exceed any buffer's ability to handle it following a long enough outage. Still, one lesson of this analysis is to use as large a buffer size on the router(s) as is economically feasible. A second lesson is to employ a more organized post-outage message-resending behavior perhaps based on a schedule or some optimization criterion. A third lesson is some kind of fail-safe communications between the router and the communications terminals, i.e., CommTerminals, that make it clear that there is an outage that is not immediately repairable. This may then trigger the scheduling protocol proposed above. Clearly, the more sophisticated schemes the router and communications terminals can employ when dealing with an outage, the more quickly and smoothly the network will recover from an extensive outage. Work has been done to mathematically model these finite buffer queues such as the one simulated in CommAspen. Reference 14 describes a

queuing network model for finite capacity queues such as is realistic in electronic finance networks.

Analysis of Economic Impact of Communications Router Congestion and Outage

One of the strengths of CommAspen is the ability to analyze the economic impact of many infrastructure incidents including communications congestion and outages. As an example of the types of analysis possible, we examined the economic impact of two different router situations. The first is the impact of router congestion over a long period of time, and the second is the impact of router outages.

Router Congestion

If a message is dropped, the router keeps trying to send the message for up to four hours of simulation time. Router delays caused by congestion can slow economic transactions in several different ways, including order delays, production delays, and late-deposit reconciliation.

Order delays happen because on a daily basis an agent's buyer places orders for all production-input commodities that are needed for the day. If it takes two hours for an order to get to the vendor, then another two hours to receive a reply about whether the order can be filled or not, a buyer can quickly run out of time to complete all ordering needs for the day. On the following day, a larger order will need to be made to replace the depleted inventory in the warehouse, plus whatever is needed to fill minimum inventory requirements. In a supply-chain economy, the effect of this inefficiency can quickly grow as large orders may require multiple vendors, i.e., multiple orderings to fill, and these vendors are also impacted by communications congestion in getting their own production inputs.

Production delays can happen when orders appear to be slow, perhaps due to a lack of communication, in which case the firm is likely to decide to produce less. When larger-than-expected orders come in from the buyers that are experiencing the order-delay problems described in the above paragraph, the seller must reply to the buyer that there is insufficient stock to fill a large order but that a smaller order can be filled. All of these communications further congest the router, slowing the communications traffic even more. Meanwhile, the seller is now receiving smaller orders which it can fill, but which deplete inventory. At some point, no further orders can be filled, and this is also reported back to the buyer through the communications router. In addition, the firm is likely to decide that the recent series of large orders require an increase in production, and the firm's buyer, in turn, makes large orders for required inputs in anticipation of increased production. These actions cascade up the supply chain causing production delays. The production and ordering impacts of a congested router occur because of overestimation of demand and the buyer's inability to meet purchasing goals because it spends too much time trying to single-source over a slow communications link.

Finally, reconciliation of bank deposits may also be delayed by slow communications. Firms can not purchase new product without sufficient funds available. Banks reconcile and recompute available funds once a day, but throughout the day new checks may be deposited. Thus some purchasing will also be delayed. This is because communication delays prevent an agent from receiving money from its customers and also prevent an agent from spending money it isn't certain that it has available.

We used two economic situations to analyze the impact of router congestion. In the first case, the economy is oversupplied with goods relative to demand; in the second case, there is a chronic shortage of goods relative to demand. It is expected that the impact of communications congestion will be greater in the undersupplied economy. The supply chain was identical in both cases except that the amount of production was varied to create an excess or shortage of goods. The input for each of the two scenarios is shown in Table 2.

Table 2. A Comparison of Demand and Production in the Excess-Supply and Supply-Shortage Scenarios Used in the Test Problem

Variable	Excess-Supply Case	Shortage Case
Consumer Demand for F	500–1,000	1,000
Maximum Production of F	1,200	900
Consumer Demand for E	100–500	500
Maximum Production of E	600	450
Maximum Demand for D	1,200	900
Maximum Production of D	1,300	880
Maximum Demand for C	3,600	2,700
Maximum Production of C	3,850	1,855
Maximum Demand for B	10,300	5,470
Maximum Production of B	11,000	5,000
Maximum Demand for A	10,100	5,510
Maximum Production of A	11,000	5,000

Because of the stochastic nature of the agent simulation, it is valuable to evaluate a number of runs to analyze the effect of parameter variations. For all of the examples discussed in the economic impact section, we averaged the results from 10 separate runs with different initial random number seeds to obtain an understanding of the impact of communication on the economy. To simulate router congestion, we used buffer size as the controlling variable. We found that a buffer size of 14 had no lost packets for this test problem and used it as the base case for the analysis. We found that a buffer size of 7 always resulted in lost packets and used it to represent the situation of a congested router.

Results are plotted in Figure 33 through Figure 35 for Firm_A, Firm_D, and Firm_F, respectively. Depicted in each graph is the cumulative amount of product in units sold by the firm in the congested case less the amount of product sold in the base case. Thus, the graph shows the impact of router congestion on the economy. We chose the amount of

product sold as a key descriptive variable rather than the value of the product because the firms would change price under the shortage conditions imposed by the router congestion and because a direct comparison to the value of product sold in an economy with no communications congestion was not possible.

The results demonstrate that in the long run the impact of chronic communications congestion has a negative effect on the economy for both the excess-supply scenario and the shortage scenario. However, as expected, the impact of communications congestion was more strongly felt in the supply-shortage economy.

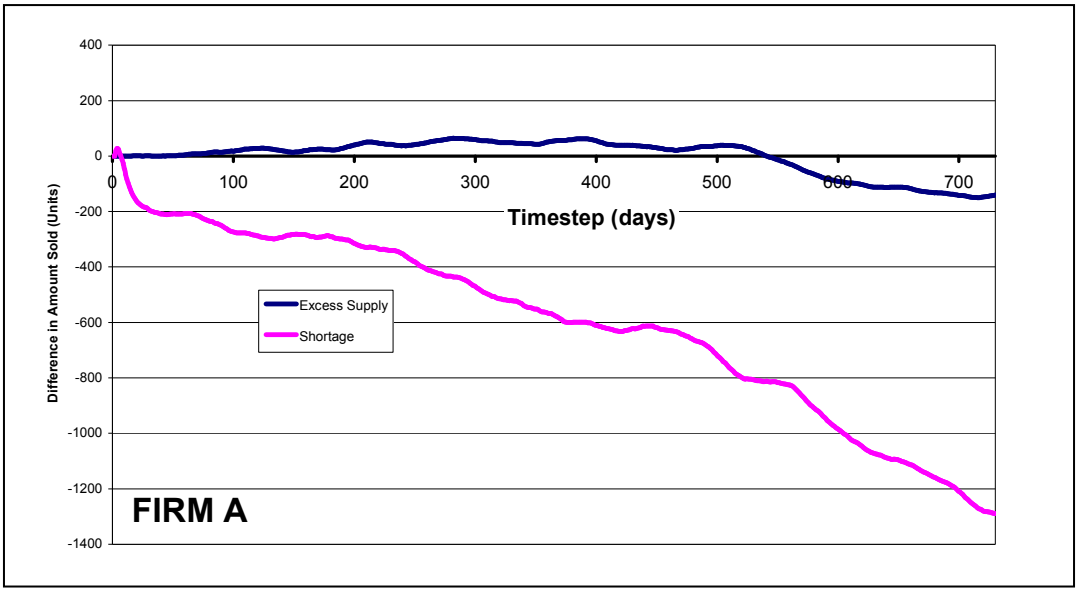


Figure 33. The average number of units sold by Firm_A-type agents under congested router conditions less the base case or noncongested router conditions. Results are shown for the excess-supply scenario and the shortage scenario.

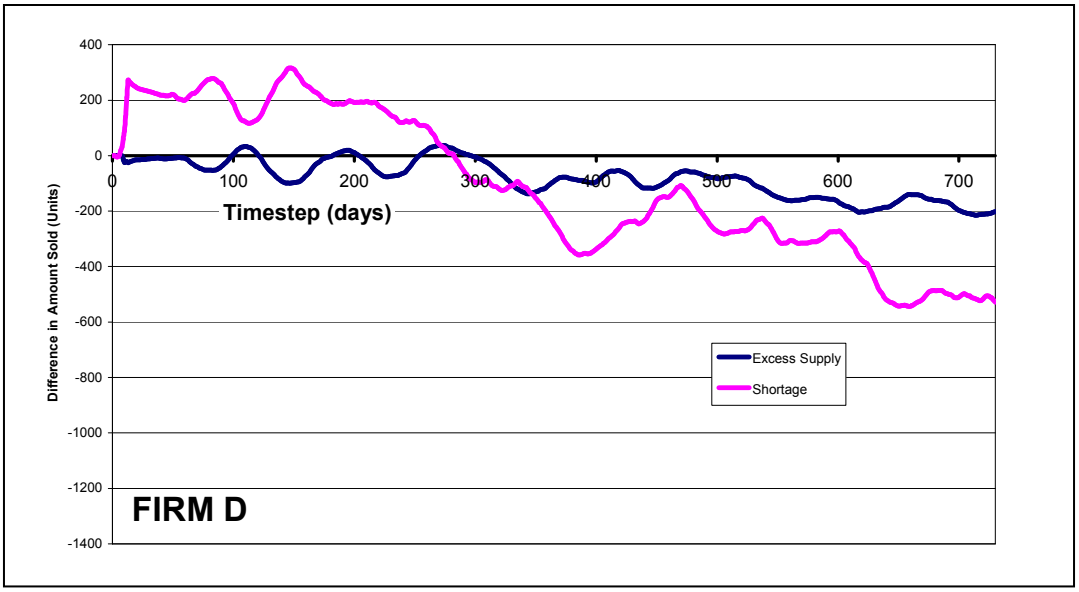


Figure 34. The average number of units sold by Firm_D-type agents under congested router conditions less the base case or noncongested router conditions. Results are shown for the excess-supply scenario and the shortage scenario.

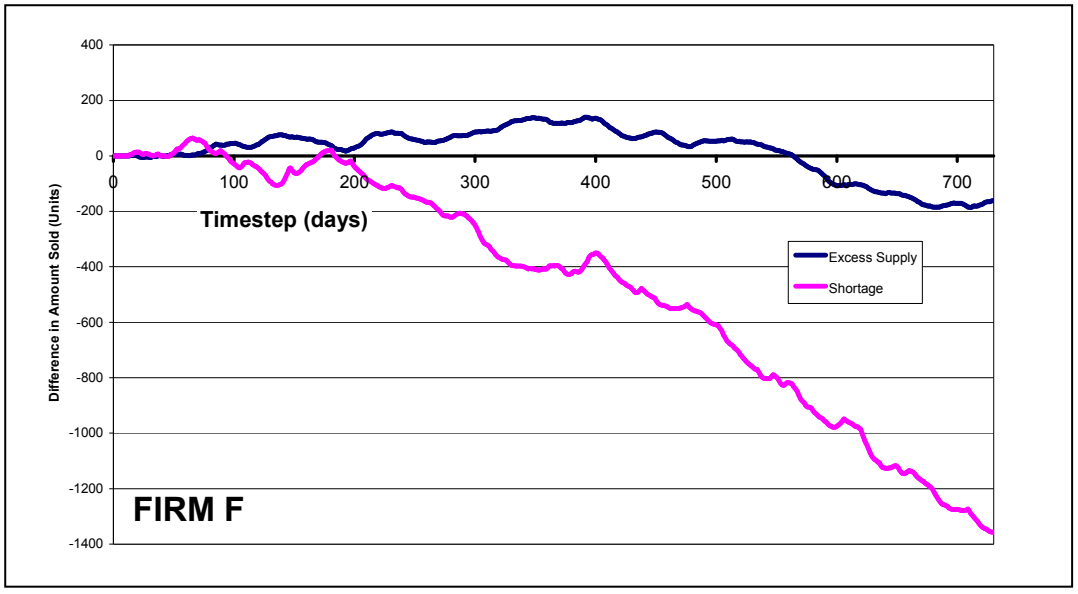


Figure 35. The average number of units sold by Firm_F-type agents under congested router conditions less the base case or noncongested router conditions. Results are shown for the excess-supply scenario and the shortage scenario.

Even the small supply-chain problem used in this test case has a great deal of complexity in how the communications congestion becomes manifested in economic efficiency of the economy. Figure 36 shows the average daily production of all F-type firms for the congested scenario less the average daily production of F-type firms for the base-case run. The graph shows the percentage of production that is lost due to communications congestion for the excess-supply and the shortage-supply scenarios versus time. The graph shows a significant loss in production for the communications-

congested case relative to the base case, although in this situation a greater level of production is lost in the excess-inventory scenario relative to the supply-shortage scenario. A closer investigation reveals that the ultimate cause of the loss of sales is a result of production inefficiency caused by input-ingredient shortages that occur when communications are congested. Figure 37 is a graph of the average amount of ingredient A that is not available to Firm_F-type agents under congested router conditions less the base case or noncongested router conditions. The graph illustrates that a lack of input ingredients is a strong factor in the production inefficiency observed during the congested communications scenario.

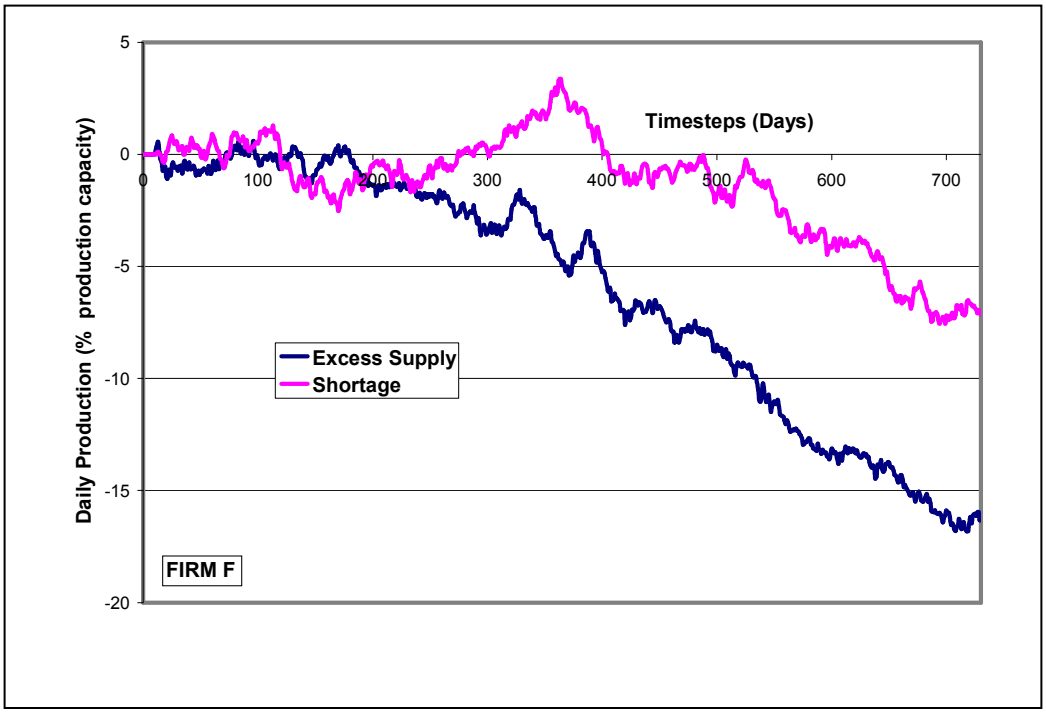


Figure 36. The average daily production of Firm_F-type agents under congested router conditions less the base case or noncongested router conditions. Results are shown for the excess-supply scenario and the shortage scenario.

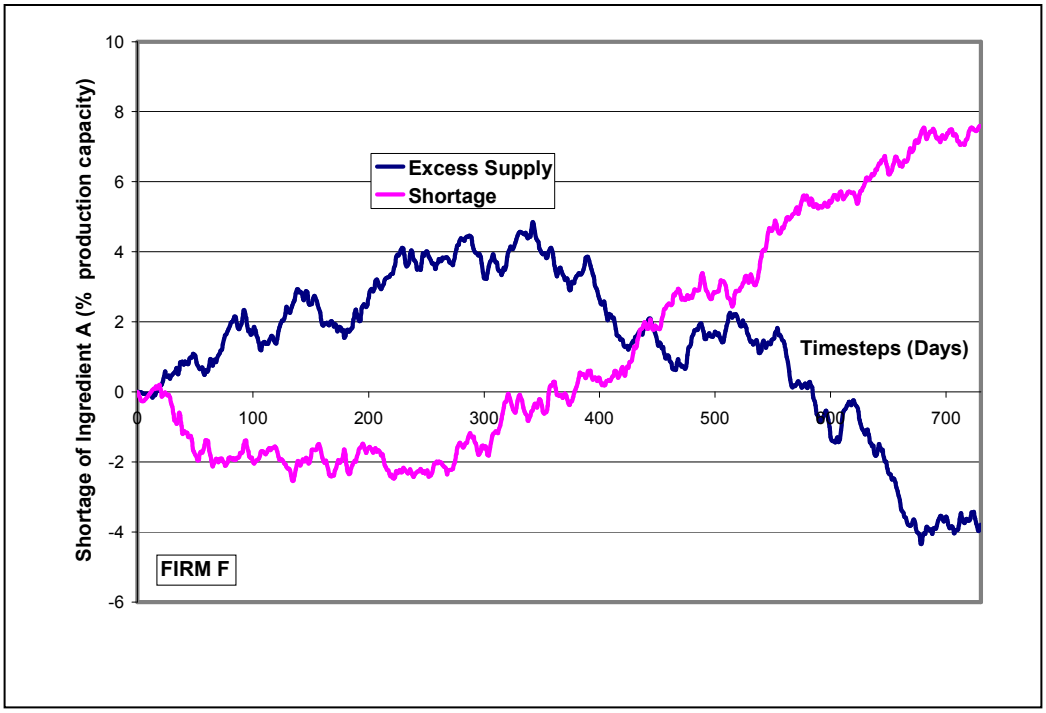


Figure 37. The average amount of ingredient A that is available to Firm_F-type agents under congested router conditions less the base case or noncongested router conditions. Results are shown for the excess-supply scenario and the shortage scenario.

Router Outage

To evaluate the impact of a communications outage on the economy, the router was stopped for a two-week outage. During this time, messages were not transmitted. Results from 10 averaged simulation runs of the two-week outage are shown in Figure 38. The difference in the amount of product sold for the outage case versus the base case is shown for the average of all firms of the same type versus time. The outage occurs between day 35 and day 49 of the simulation. The results show that the outage results in a loss of sales for all firm types relative to the base case for a long period of time following the outage. Although a new equilibrium is re-established by all of the firms, it takes approximately 150 days following the outage. After this time, sales reach a steady state relative to the base case.

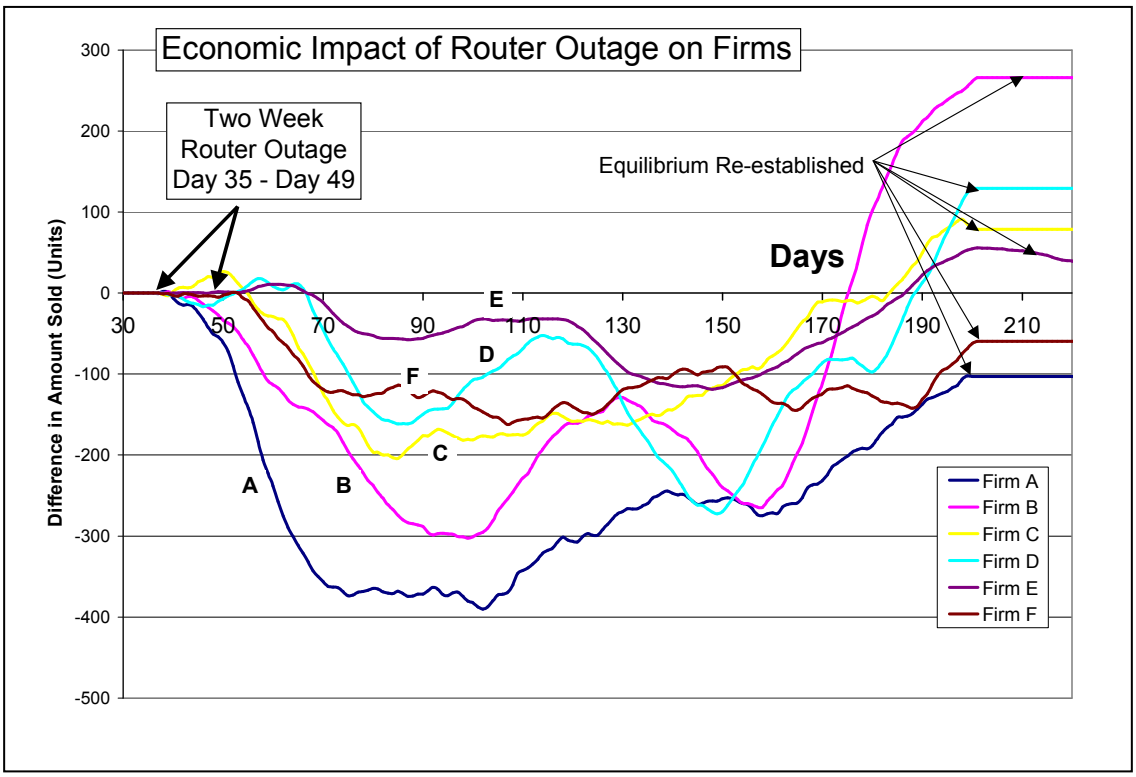


Figure 38. The average amount of units sold by each firm type under outage conditions less the base-case conditions. The router outage occurs between day 35 and day 49 of the simulation.

The economic analysis runs presented here indicate that communications inefficiencies and outages have a direct correlation to the economic efficiency of agents in the simulated supply-chain economy just as they do in the real world. CommAspen could be used to evaluate how different communications policies that address banking communications practices could impact the economy.

Applications and Future Plans

Analysis of results for the sample scenarios points out how CommAspen can be applied to examine the economic costs of disruptions in the telecommunications infrastructure. CommAspen represents our latest effort at developing a tool that allows users to investigate the impact of telecommunications outages on other infrastructures. Being an economic model, CommAspen focuses on the economic impact of an outage. When compared to other infrastructure interdependency models, CommAspen pushes the envelope of what is possible in investigating communication traffic.

As with Aspen-EE, CommAspen can also be configured to explore more widespread vulnerabilities in the economy such as shifts in product and labor markets that might occur when governmental policies coupled with generation company choices result in disruptions such as power outages. Such analyses can be run before any such outage event occurs and provide calculated results of losses that could occur.

Expanding on the research performed in this LDRD, CommAspen has been upgraded with a number of enhancements under a contract funded by the National Infrastructure Simulation and Analysis Center (NISAC). This center is a partnership between Los Alamos National Laboratory and Sandia that was formally chartered by Congress in 2001 [15]. New features added under the NISAC contract include a graphical user interface, better graphical output than CommAspen, a transportation/shipping infrastructure, and the ability to communicate with agent models developed at other research institutions. CommAspen with the new features is called N-ABLE. See Reference 16 for a description of the N-ABLE software package.

We also envision a number of other enhancements to CommAspen/N-ABLE. One enhancement is to develop a user interface that supports preparation of the input file, rather than requiring the input to be created via a text editor. A second enhancement is to create a parallel-processor version of the model. A parallel implementation would run faster and enable us to better address large-scale problems. Adding other infrastructures like water is a third enhancement that would facilitate examination of the interdependency between more infrastructure networks. Using our spigot concept, the addition of other infrastructures is relatively simple. A fourth enhancement would be to develop additional software tools to take real-world databases, such as thousands of firms in an area, and turn these firms into agents, allowing us to quickly develop studies for a particular locality. Currently, this type of activity is an intensely manual process.

And while the communications network in the study answered the questions of immediate concern, we also intend to develop a much more sophisticated communications network. This network could include, for example, multiple routers versus the single router currently in CommAspen. Such a network could also include a backup type of agent, so that we could examine how backup systems in telecommunications could help to divert disruptions in service. In addition, we have laid the groundwork to model specific types of communications networks for different media such as paging systems, copper, and fiber optics. Further work needs to be done to develop agents with this level of detail.

References

1. Sandia National Laboratories. *U.S. Infrastructure Assurance Strategic Roadmaps: Strategies for Preserving Our National Security*, SAND98-1496. Albuquerque, NM: Sandia National Laboratories, 1998.

2. U.S. Department of Homeland Security. *The National Strategy for the Physical Protection of Critical Infrastructures and Key Assets*. February 2003. Available at http://www.dhs.gov/interweb/assetlibrary/Physical_Strategy.pdf (accessed 13 September 2003).

3. Barton, D. C., and K. L. Stamber. "An Agent-Based Microsimulation of Critical Infrastructure Systems." In *Global Energy Exposition 2000 Proceedings*, held in Las Vegas, NV, July 2000. Lancaster, PA: Technomic Publishing Company, 2000.

4. Barton D. C., E. D. Eidson, D. A. Schoenwald, K. L. Stamber, and R. K. Reinert. *Aspen-EE: An Agent-Based Model of Infrastructure Interdependency*, SAND2000-2925. Albuquerque, NM: Sandia National Laboratories, 1998.

5. Cox, R. G., and R. K. Reinert. *A Year 2003 Conceptual Model for the U.S. Telecommunications Infrastructure*, SAND number pending at time of publication. Albuquerque, NM: Sandia National Laboratories, October 2003.

6. Basu, N., R. J. Pryor, T. Quint, and T. Arnold. *Aspen: A Microsimulation Model of the Economy*, SAND96-2459. Albuquerque, NM: Sandia National Laboratories, October 1996.

7. Ehlen, M. A. *Search Costs, Strategic Pricing and Market Stability: The Importance of Firm Learning on Equilibrium Price Dispersion* (Draft). Albuquerque, NM: Sandia National Laboratories, September 2003.

8. Messmer, E. "Protecting the Financial Infrastructure." *NetworkWorldFusion*, 2 June 2003. Available at http://www.nwfusion.com/weblogs/security/002856.html (accessed 25 June 2003).

9. Tritak, J. S. *Statement of John S. Tritak, Director, Critical Infrastructure Assurance Office, Bureau of Industry and Security, before the House Committee on Government Reforms Subcommittee on Government Efficiency, Financial Management and Intergovernmental Relations*. U.S. Department of Homeland Security. Testimony July 24, 2002. Available at http://www.ciao.gov/publicaffairs/tritak7.24:02.html (accessed 25 June 2003).

10. *Summary of "Lessons Learned' from Events of September 11 and Implications for Business Continuity*. U.S. Securities and Exchange Commission. 13 February 2002. Available at http://www.sec.gov/divisions/marketreg/lessonslearned.htm (accessed 25 June 2003).

11. Federal Reserve System. *Interagency Paper on Sound Practices to Strengthen the Resilience of the U.S. Financial System*. [Docket No. R-1128]. 7 April 2003. Available at http://www.federalreserve.gov/boarddocs/press/bcreg/2003/20030408/attachment.pdf (accessed 04 September 2003).

12. Blumenfeld, L. "Dissertation Could Be Security Threat." *Washingtonpost.com*, 8 July 2003. Available at http://www.washingtonpost.com/ac2/wp-dyn/A23689-2003Jul7?language=printer (accessed 09 September 2003).

13. "RFC INDEX." Available at http://rfc-1232.rfcindex.com/rfc-1232-5.htm (accessed 3 October 2003).

14. Balsamo, S., V. De Nitto Personè, and P. Inverardi. "A Review on Queueing Network Models with Finite Capacity Queues for Software Architectures Performance Prediction." *Performance Evaluation* 974 (2002): 1–20.

15. National Infrastructure Simulation and Analysis Center. Available at http://www.sandia.gov/CIS/NISAC.htm (accessed 20 August 2003).

16. Ehlen, M., and E. Eidson. *NISAC Agent-Based Laboratory for Economics (N-ABLE): Agent and Simulations Architecture* (Draft). Albuquerque, NM: Sandia National Laboratories, September 2003.

Appendix A: Input File Description

This appendix describes the format and contents of the user-prepared input file to CommAspen. The appendix also includes reference information that is necessary for defining parameters in several parts of the CommAgent specification.

File Format

The CommAspen Input File is prepared by the user in XML. The file, as described in this report consists of three sections: XML Document Recommendation, General Comments, and MODEL Data. The MODEL Data section has three subsections: Run, CommAgent, and Problem Data. Figure A-1 gives a general overview of the file format.

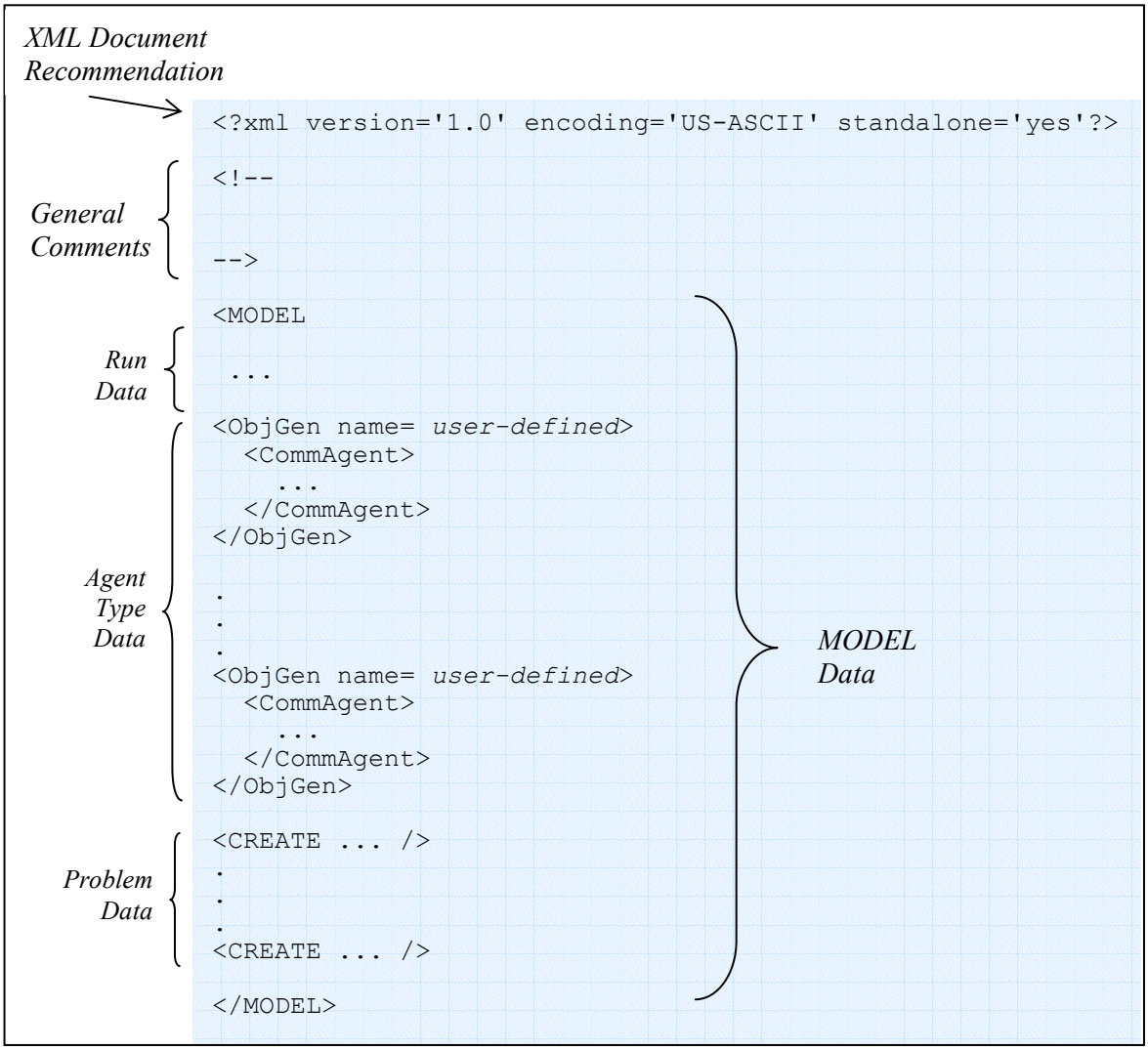


Figure A-1. Organization of the CommAspen Input File.

Input Conventions

Specification of the input data follows the rules of XML, where what we have termed blocks, items, and statements are referred to in XML as elements, and what we have termed parameters are referred to in XML as attributes. Thus, the input formats for elements and attributes follow XML syntax. Please refer to an XML reference for information on the structure of an XML document.

Some general pointers for preparing or interpreting the CommAspen Input File are as follows:

- Each element (what we refer to as a block, an item, or a statement) begins with a start tag and ends with an end tag.

- The start tag can be specified in one of two forms:

  <*element_name* as in `<InitialFunds`

  or

  <*element_name*> as in `<InitialFunds>`

- An element may contain other elements or attributes (what we refer to as parameters). Each attribute is specified in the form *name*='*value'*, as in the following:

```
<InitialFunds
   amount='10000000'
</InitialFunds>
```

- Attribute values are set off by pairs of single quotes (as above) or double quotes, based on the user's preference.

- White space can be used where necessary for indentation, as well as to make input data more readable, e.g., amount = '10000000', where white space has been inserted before and after the equals sign.

- The end tag can be specified in one of two forms:

  </*element_name*> as in `</InitialFunds>`

  or

  /> as in />

So following the second form of the end tag and deciding not to indent the attribute, we could place the entire block on a single line.

```
<InitialFunds amount='10000000' />
```

File Contents

The contents of the user-prepared input file to CommAspen are described below. Examples of initial values are given for all parameters listed. Units are specified in the description column of the tables. Note that parameter entries (or attributes as they are called in XML) are required in the input format of *name*='*value'*, though that is not the exact format followed in these tables.

Section One – Xml Document Recommendation

The first section of the input file contains parameters that are part of the XML document recommendation. This section begins with <? and ends with ?>.

Name	Description	Example Value
xml version	Version identifier. Note that only '1.0' is valid at this time.	'1.0'
encoding	Machine code used to encode the characters. Conforming XML processors must be able to read encodings of UTF-8 and UTF-16. The example here is a subset of UTF-8.	'US-ASCII'
standalone	Dependence of input file on other documents. Valid values are 'yes' and 'no'.	'yes'

Section Two – General Comments

This section of the input file contains general comments about the specific run. No user-input parameters are used in this section. This section is not required, but it is useful. See Appendix B for examples of inserted comments.

Section Three – MODEL Data

This section of the input file contains information about the run, the CommAgents, and the problem. Parameters within the section are described below.

Run Data

This part of the MODEL data contains general information about the run.

Name	Description	Example Value
PauseTime	Total number of time steps, i.e., minutes in the problem	'1051200'
SnapshotFrequency	Edit frequency for printing data to an output file	'1440'
ProgressIndication	Frequency at which the model prints out the current time step. Units are in multiples of time steps, e.g., a value of '60' directs the model to print out every 60th time step (0, 60, 120, 180, etc.).	'60'
RandomSeed	Random number seed	'65535'
Description	Any user-specified information that describes the problem	'2_CommTest'
SimStreamServer	Internet Protocol address, or host name	'nisac-srn.sandia.gov'
SimStreamPort	Port number	'998'
BankRouterBUFSIZE	Buffer size of the global router used in simulating financial transactions in the communications network. This is the number of packets the router can hold. Currently, a packet corresponds to a message.	'20'
SimulationID	Not currently used, but requires a value	'9'

Agent Type Data

This part of the MODEL data contains information about all the types of CommAgents in the problem. Any number of agent types may be specified. Each agent type is specified in an ObjGen block. Values for numerical parameters can be specified either as a single value or as a range. Ranges allow CommAspen to create agents that have some variations instead of being identical copies. Figure A-2 gives an overview of the organization of data for each type of CommAgent.

```
<ObjGen name= user-defined>
  <CommAgent>
    <Location              A maximum of one Location block is allowed.
      ... />
    <InitialFunds          A maximum of one InitialFunds block is allowed.
      ... />
    <Productions>          A maximum of one Productions block is allowed..
       <type-of-production     There must be at least one type-of- production block
                                per Productions block..
          ... />
    </Productions>
    <Buyers>               A maximum of one Buyers block is allowed.
      <type_of_buyer            There must be at least one type-of-buyers block per
                                Buyers block.
        ... />
    </Buyers>
    <InitialStock>         A maximum of one  InitialStock block is allowed.
      ... />

  </CommAgent>
</ObjGen>
```

Figure A-2. General organization of a CommAgent specification.

Given that there are a variety of other blocks, statements, and items available to construct the agent types we have built for the problem in this report, it is important to provide a summary for the particular blocks to which these agents have access. This summary could be particularly helpful in deciphering which of the type-of-production, type-of-seller, and type-of-buyer blocks are available to each type (or types) of agents. Thus, in Table A-1, we show an abbreviated summary. Note that some of these components do not appear in the sample problem for particular types of agents.

Table A-1. Valid Names for Agent Types

Bank	Consumer	All Firm Agent Types
<Location> <InitialFunds> <Productions> <BankAccountProduction> <DAILY_CONSUMPTION> <CONSUME> <DISTRIBUTION> <Sellers> <BankAccountSeller> <Buyers> <BankAccountBuyer> <InitialStock> <Stock>	<Location> <InitialFunds> <Productions> <Consumer> <DAILY_CONSUMPTION> <CONSUME> <DISTRIBUTION> <Buyers> <BankAccountBuyer> <FirmBuyer> <InitialStock> <Stock>	<Location> <InitialFunds> <Productions> <FirmProduction> <Sellers> <FirmSeller> <INPUTS> <INPUT> <Buyers> <BankAccountBuyer> <FirmBuyer> <InitialStock> <Stock>

Following is a description of the contents of a CommAgent specification.

ObjGen Block

An ObjGen block encapsulates the specification of a type of CommAgent. There is a single parameter in this block, followed immediately by an embedded CommAgent block.

ObjGen Block Name	Description	Example Value
name	User-defined name of agent type	'FIRM_A'

CommAgent Block

A CommAgent contains five major components, or blocks. Each block is described below.

Location Block

Data pertaining to the location of an agent are specified in the Location block.

Location Block Name	Description	Example Value
latitude	Initial latitude in *XYZ* coordinate system, in miles	'0 - 1000'
longitude	Initial longitude in *XYZ* coordinate system, in miles	'0 - 1000'
elevation	Initial elevation in *XYZ* coordinate system, in miles	'0.0'

InitialFunds Block

Data pertaining to the starting cash assets assigned to a CommAgent is specified in the InitialFunds block.

InitialFunds Block Name	Description	Example Value
amount	Initial amount of starting cash, in dollars	'1000000'

Productions Block

The Productions block can contain zero or more *type-of-production* blocks. Rarely, though, would any such blocks be specified. There are currently three valid blocks for the type-of-production: BankAccountProduction, FirmProduction, and Consumer. For ease of use, the contents of each of these blocks will be handled separately, though it is recognized that there are some components that are accessed by more than one type-of-production block.

BankAccountProduction Block

- There are currently no individual parameters describing the production for a BankAccountProduction block.

- A DAILY_CONSUMPTION block can be specified for a BankAccountProduction block, and it may be accompanied by a DISTRIBUTION block. See the description of these blocks below in the paragraph entitled Consumer Block, given that that block is currently the main user of the DAILY_CONSUMPTION block and the DISTRIBUTION block.

- There is a Sellers block in a BankAccountProduction block. The Sellers block consists of at least one BankAccountSeller block. The individual parameter for this block is listed below.

Sellers Block BankAccountSeller Block Name	**Description**	**Example Value**
region	Identification of area to which seller belongs. An alphanumeric or character string is allowed.	'1'

Consumer Block

- There are currently no individual parameters for a Consumer block.

- A DAILY_CONSUMPTION block can be specified for a Consumer block (as it can for a BankAccountProduction block). Each DAILY_CONSUMPTION block can contain one or more CONSUME items. The parameters in a CONSUME item are defined below.

DAILY_CONSUMPTION Block CONSUME Item Name	**Description**	**Example Value**
amount	Quantity of the commodity consumed daily, in units of production	'1 - 5'
commodity	Name of the commodity consumed daily	'COMPONENT_E'

- A DAILY_CONSUMPTION block can be accompanied by a DISTRIBUTION block. This block is used to specify how much of a certain consumption commodity is used by the agent on a 24-hour basis. Values are specified for this block in an array with labeled hours.

DISTRIBUTION Block Name	Description	Example Value
hour0 to hour23	A labeled array of 24 entries, where each value represents the fraction of total daily use of the commodities specified in the associated DAILY_CONSUMPTION block within the hour	'1.0' for one entry

FirmProduction Block

- The FirmProduction block has three individual parameters, as described below.

FirmProduction Block Name	Description	Example Value
initial_amount_produced_daily	Initial quantity of the commodity produced daily, in units of production	'500'
max_amount produced_daily	Maximum quantity of the commodity produced daily, in units of production	'675'
OUTPUT	Name of the commodity produced daily	'COMPONENT_A'

- There is a Sellers block in a FirmProduction block. The Sellers block consists of at least one FirmSeller block. The individual parameters for the FirmSeller block are listed below.

Sellers Block FirmSeller Block Name	Description	Example Value
region	Identification of area to which seller belongs. An alphanumeric or character string is allowed.	'1'
initial_advertised_price	Initial price of commodity, in dollars	'1.17 - 3.00'
sampling_prob	Market presence of commodity, expressed as a probability. Cumulative values for all agent types should add to 1.	'0.3333'

- An INPUTS block may be present in a FirmProduction block to specify a production recipe. The INPUTS block can contain one or more INPUT items. Together, these items specify the recipe for a product. Following are the individual parameters in an INPUT item.

INPUTS Block INPUT Item Name	Description	Example Value
commodity	Name of commodity required as part of recipe	'COMPONENT_D'
amount	Quantity of commodity needed, in units of production	'1'

Buyers Block

There can be at most one Buyers block within a CommAgent specification. The Buyers block can contain up to two type-of-buyer blocks, depending on the agent type. There are currently two valid blocks for the type of buyer: BankAccountBuyer and FirmBuyer.

BankAccountBuyer Block

- There is currently one individual parameter describing the production for a BankAccountBuyer block, as defined below.

BankAccountBuyer Block Name	Description	Example Value
region	Identification of area to which buyer belongs. An alphanumeric or character string is allowed.	'1'

FirmBuyer Block

- There are currently five individual parameters in a FirmBuyer block, as defined below.

FirmBuyer Block Name	Description	Example Value
region	Identification of area to which buyer belongs. An alphanumeric or character string is allowed.	'1'
commodity	Name of commodity to purchase	'COMPONENT_B'

FirmBuyer Block Name	Description	Example Value
order_chunk	Quantity of commodity to purchase at one time, in units of production	'1.0'
max_acceptable_price	Maximum acceptable amount to pay for commodity, in dollars	'10.0'
search_cost	Cost in time, money, and effort for a single sample of the market, i.e., to go to one vendor and discover its price. Higher values indicate each sample requires more from the agent. As a result, the agent performs fewer samples.	'5.0'

InitialStock Block

This block is composed of one or more Stock items. Following are the individual parameters in a Stock item.

InitialStock Block Stock Item Name	Description	Example Value
commodity	Name of commodity to stock in the warehouse	'COMPONENT_D'
amount	Initial quantity of commodity, in units of production	'100'
max_capacity	Maximum amount the warehouse can store of the commodity, in units of production	'1000'
cost_per_unit	Average cost of each unit of the commodity stored in the warehouse, in dollars. This cost is used to compute the production cost and/or marginal cost for profits	'5.0'

Problem Data

This part of the MODEL data identifies the types and associated numbers of individual agents to create of the defined types for the run. The section consists of

CREATE statements, one for each type of CommAgent that will be created in the run. A description of the parameters in a CREATE statement follows.

Create Statement Name	Description	Example Value
name	Name of agent type, from the list of ObjGen block names specified by the user	'FIRM_A'
quantity	Number of agents of this type to create	'5'

Reference Information

This part of the appendix describes information that is common to one or more parameters in the CommAgent Data specification.

Predefined Values

A number of components and subcomponents of a CommAgent definition have an input parameter that specifies the name of a commodity, or product. These names are predefined in the model code. The list of acceptable values for the commodity parameter follows. Note that the list can be updated by CommAspen developers for new kinds of products.

For Bank Only:

```
BANK_SAVINGS
```

All Other Commodities:

```
NUTS_IN          GOODS
NUTS_OUT         INSTRU
BOLTS_IN         LABOR
BOLTS_OUT        MACH
COMPONENT_A      PLANE
COMPONENT_B      POWER
COMPONENT_C      EMCASPOWER
COMPONENT_D      TELCO
CAPITAL          TRANSPORT
ELEC             UTILITY
FABMET
```

Units of Production

Some of the parameters in a CommAgent specification describe the units as "units of production." Each such unit is expressed as a fraction of a dollar of inputs to produce one dollar of output. Thus, for example, to make $1 of CAR, you might need $0.10 of rubber, $0.05 unit of plastic, $0.25 of steel, etc. All units of production are thus expressed in dollars, or monetary units. Much of the data input to CommAspen is received in the form

of economic input/output tables. Everything in these tables is expressed in economic dollars.

Appendix B: Sample Input File

Following is an example of the CommAspen Input File that was used for simulating the problem in this report.

```
<?xml version='1.0' encoding='US-ASCII' standalone='yes'?>

<!--
  Aspen 5.0 Model Specification File.

  N-ABLE Content

  The Model Specification File allows the Model Generator to create
  and initialize a model, then store that model to a Restart file.
  Aspen 5.0 Models obtain all input for execution from Restart files
  (which are also XML files that can be edited by users).

  -->

<!-- ***************************** -->

<!--
  CommAspen works with a timestep resolution of 1-minute;
   we adapt figures from ShockAspen's 1-hour resolution

  84 days == 2016 hours == 120960 minutes
   7 days ==  168 hours == 10080 minutes
   1 day  ==   24 hours == 1440 minutes

  PauseTime       = '120960'
  ProgressIndication    = '1'
-->

<MODEL
  PauseTime       = '1051200'
  SnapshotFrequency     = '1440'
  ProgressIndication    = '60'
  RandomSeed            = '65535'
  Description           = '2_CommTest'
  SimStreamServer = 'nisac-srn.sandia.gov'
  SimStreamPort         = '9998'

  BankRouterBUFSIZE     = '20'

  SimulationID          = '9'>

<!-- ***************************** -->
<!-- Define Banks -->

<ObjGen name='Bank'>
  <CommAgent>
    <Location
      latitude='0 - 10000' longitude='0 - 10000' elevation='0.0'/>
```

```
    <InitialFunds amount='0'/>

    <Productions>
      <BankAccountProduction>
        <!--
      <DAILY_CONSUMPTION>
            <CONSUME amount = '1' commodity = 'COMPONENT_A' />
        </DAILY_CONSUMPTION>
      -->

        <Sellers>
          <BankAccountSeller region='1'/>
        </Sellers>

      </BankAccountProduction>
    </Productions>

    <Buyers>
      <BankAccountBuyer region='1'/>
    </Buyers>

  </CommAgent>
</ObjGen>


<!-- **************************** -->
<!-- Define Firms -->

<ObjGen name='FIRM_A'>
  <CommAgent>
    <Location
      latitude='0 - 10000'
      longitude='0 - 10000'
      elevation='0.0'/>

    <InitialFunds amount='1000000'/>

    <Productions>
        <FirmProduction initial_amount_produced_daily='500'
                        max_amount_produced_daily='675'
                        OUTPUT='COMPONENT_A'>
          <Sellers>
            <FirmSeller
            region='1'
            initial_advertised_price='1.17 - 3.00'
            sampling_prob='0.3333'/>
          </Sellers>
        </FirmProduction>
    </Productions>

    <Buyers>
      <BankAccountBuyer region='1'/>
    </Buyers>

    <InitialStock>
      <Stock commodity='COMPONENT_A' amount='100' max_capacity='1000'
cost_per_unit='5.00'/>
    </InitialStock>

  </CommAgent>
```

```
</ObjGen>

      <!-- -->
<ObjGen name='FIRM_B'>
  <CommAgent>
    <Location
      latitude='0 - 10000'
      longitude='0 - 10000'
      elevation='0.0'/>

    <InitialFunds amount='1000000'/>

    <Productions>
        <FirmProduction initial_amount_produced_daily='400'
                        max_amount_produced_daily='450'
                        OUTPUT='COMPONENT_B'>
          <Sellers>
            <FirmSeller
            region='1'
            initial_advertised_price='1.25 - 2.00'
            sampling_prob='0.3333'/>
          </Sellers>
        </FirmProduction>
    </Productions>

    <Buyers>
      <BankAccountBuyer region='1'/>
    </Buyers>

    <InitialStock>
      <Stock commodity='COMPONENT_B' amount='100' max_capacity='1000'
cost_per_unit='5.00'/>
    </InitialStock>

  </CommAgent>
</ObjGen>

<!-- -->

<ObjGen name='FIRM_D'>
  <CommAgent>
    <Location
      latitude='0 - 10000'
      longitude='0 - 10000'
      elevation='0.0'/>

    <InitialFunds amount='1000000'/>

    <Productions>
        <FirmProduction initial_amount_produced_daily='300'
                        max_amount_produced_daily='325'
                        OUTPUT='COMPONENT_D'>
          <Sellers>
            <FirmSeller
            region='1'
            initial_advertised_price='2.50 - 4.50'
            sampling_prob='0.3333'/>
          </Sellers>
          <INPUTS>
            <INPUT commodity = 'COMPONENT_B'  amount = '1'/>
        </INPUTS>
```

```
        </FirmProduction>
    </Productions>

    <Buyers>
      <BankAccountBuyer region='1'/>
      <FirmBuyer
            region='1'
            commodity='COMPONENT_B'
            order_chunk='1.0'
            max_acceptable_price='10.0'
      search_cost='5.00' />
    </Buyers>

    <InitialStock>
      <Stock commodity='COMPONENT_D' amount='100' max_capacity='1000'
cost_per_unit='5.00'/>
      <Stock commodity='COMPONENT_B' amount='100' max_capacity='1000'
cost_per_unit='5.00'/>
    </InitialStock>

  </CommAgent>
</ObjGen>

<!-- -->

<ObjGen name='FIRM_C'>
  <CommAgent>
    <Location
      latitude='0 - 10000'
      longitude='0 - 10000'
      elevation='0.0'/>

    <InitialFunds amount='1000000'/>

    <Productions>
        <FirmProduction initial_amount_produced_daily='600'
                        max_amount_produced_daily='650'
                        OUTPUT='COMPONENT_C'>
          <Sellers>
            <FirmSeller
            region='1'
            initial_advertised_price='3.50 - 5.50'
            sampling_prob='0.3333'/>
          </Sellers>
          <INPUTS>
            <INPUT commodity = 'COMPONENT_A'  amount = '1'/>
            <INPUT commodity = 'COMPONENT_B'  amount = '1'/>
        </INPUTS>
        </FirmProduction>
    </Productions>

    <Buyers>
      <BankAccountBuyer region='1'/>
      <FirmBuyer
            region='1'
            commodity='COMPONENT_A'
            order_chunk='1.0'
            max_acceptable_price='10.0'
      search_cost='5.00' />
      <FirmBuyer
            region='1'
```

```
            commodity='COMPONENT_B'
            order_chunk='1.0'
            max_acceptable_price='10.0'
      search_cost='5.00' />
    </Buyers>

    <InitialStock>
      <Stock commodity='COMPONENT_C' amount='100' max_capacity='1000'
cost_per_unit='5.00'/>
      <Stock commodity='COMPONENT_A' amount='100' max_capacity='1000'
cost_per_unit='5.00'/>
      <Stock commodity='COMPONENT_B' amount='100' max_capacity='1000'
cost_per_unit='5.00'/>
    </InitialStock>

  </CommAgent>
</ObjGen>

<!-- -->

<ObjGen name='FIRM_E'>
  <CommAgent>
    <Location
      latitude='0 - 10000'
      longitude='0 - 10000'
      elevation='0.0'/>

    <InitialFunds amount='1000000'/>

    <Productions>
        <FirmProduction initial_amount_produced_daily='100'
                        max_amount_produced_daily='125'
                        OUTPUT='COMPONENT_E'>
          <Sellers>
            <FirmSeller
            region='1'
            initial_advertised_price='3.50 - 7.50'
            sampling_prob='0.3333'/>
          </Sellers>
          <INPUTS>
            <INPUT commodity = 'COMPONENT_D'  amount = '1'/>
            <INPUT commodity = 'COMPONENT_C'  amount = '1'/>
        </INPUTS>
        </FirmProduction>
    </Productions>

    <Buyers>
      <BankAccountBuyer region='1'/>
      <FirmBuyer
            region='1'
            commodity='COMPONENT_C'
            order_chunk='1.0'
            max_acceptable_price='10.0'
      search_cost='5.00' />
      <FirmBuyer
            region='1'
            commodity='COMPONENT_D'
            order_chunk='1.0'
            max_acceptable_price='10.0'
      search_cost='5.00' />
    </Buyers>
```

```
    <InitialStock>
      <Stock commodity='COMPONENT_E' amount='100' max_capacity='1000'
cost_per_unit='5.00'/>
      <Stock commodity='COMPONENT_C' amount='100' max_capacity='1000'
cost_per_unit='5.00'/>
      <Stock commodity='COMPONENT_D' amount='100' max_capacity='1000'
cost_per_unit='5.00'/>
    </InitialStock>

  </CommAgent>
</ObjGen>

      <!-- -->

<ObjGen name='FIRM_F'>
  <CommAgent>
    <Location
      latitude='0 - 10000'
      longitude='0 - 10000'
      elevation='0.0'/>

    <InitialFunds amount='1000000'/>

    <Productions>
        <FirmProduction initial_amount_produced_daily='50'
                        max_amount_produced_daily='250'
                        OUTPUT='COMPONENT_F'>
          <Sellers>
            <FirmSeller
            region='1'
            initial_advertised_price='4.50 - 5.50'
            sampling_prob='0.3333'/>
          </Sellers>
          <INPUTS>
            <INPUT commodity = 'COMPONENT_A'  amount = '1'/>
            <INPUT commodity = 'COMPONENT_C'  amount = '1'/>
        </INPUTS>
        </FirmProduction>
    </Productions>

    <Buyers>
      <BankAccountBuyer region='1'/>
      <FirmBuyer
            region='1'
            commodity='COMPONENT_A'
            order_chunk='1.0'
            max_acceptable_price='10.0'
      search_cost='5.00' />
      <FirmBuyer
            region='1'
            commodity='COMPONENT_C'
            order_chunk='1.0'
            max_acceptable_price='10.0'
      search_cost='5.00' />
    </Buyers>

    <InitialStock>
      <Stock commodity='COMPONENT_F' amount='100' max_capacity='1000'
cost_per_unit='5.00'/>
```

```
      <Stock commodity='COMPONENT_C' amount='100' max_capacity='1000'
cost_per_unit='5.00'/>
      <Stock commodity='COMPONENT_A' amount='100' max_capacity='1000'
cost_per_unit='5.00'/>
    </InitialStock>

  </CommAgent>
</ObjGen>

<!-- ************************** -->
<!-- Define Consumers -->

<ObjGen name='Consumer'>
  <CommAgent>
    <Location
     latitude='0 - 10000'
      longitude='0 - 10000'
      elevation='0.0'/>

    <InitialFunds amount='2000000'/>

    <Productions>
        <Consumer>
          <DAILY_CONSUMPTION>
            <CONSUME amount = '5-10' commodity = 'COMPONENT_F' />
            <CONSUME amount = '1-5' commodity = 'COMPONENT_E' />
          </DAILY_CONSUMPTION>
        </Consumer>
    </Productions>

    <Buyers>
      <FirmBuyer
        region='1'
        commodity='COMPONENT_F'
        order_chunk='1.0'
        max_acceptable_price='10'
        search_cost='5.00'/>
      <FirmBuyer
        region='1'
        commodity='COMPONENT_E'
        order_chunk='1.0'
        max_acceptable_price='10'
        search_cost='5.00'/>
      <BankAccountBuyer region='1'/>
    </Buyers>

    <InitialStock>
      <Stock commodity='COMPONENT_E' amount='50' max_capacity='1000'
cost_per_unit='1.00'/>
      <Stock commodity='COMPONENT_F' amount='50' max_capacity='1000'
cost_per_unit='1.00'/>
    </InitialStock>

  </CommAgent>
</ObjGen>



<!-- ************************** -->

<CREATE name='Bank' quantity='1'/>
```

```
<CREATE name='FIRM_A' quantity='5'/>
<CREATE name='FIRM_B' quantity='6'/>
<CREATE name='FIRM_C' quantity='3'/>
<CREATE name='FIRM_D' quantity='2'/>
<CREATE name='FIRM_E' quantity='5'/>
<CREATE name='FIRM_F' quantity='5'/>

<CREATE name='Consumer' quantity='100'/>

</MODEL>
```

Distribution

1	0310	R. Leland	09220
1	0318	J. E. Nelson	09216
5	0318	D. C. Barton	09216
5	0318	D. A. Schoenwald	09216
1	0196	B. N. Chenoweth	09216
1	0196	J. A. Sprigg	09216
1	0847	D. J. Melander	09226
1	0316	A. Slepoy	09235
1	0318	C. R. Jorgensen	09216
1	0318	M. B. E. Boslough	09216
1	1231	T. J. Allard	00080
1	1207	J. R. Yoder	05935
1	1201	J. R. Gosler	05004
1	0318	P. Yarrington	09230
17	0318	S. G. Wagner	09209
1	0321	W. J. Camp	09200
1	0323	D. L. Chavez	01030
1	0451	S.G. Varnado	06500
1	0451	S. Rinaldi	06541
1	0451	M. Ehlen	06541
5	0451	E. D. Eidson	06541
1	0748	M. S. Allen	06413
5	0748	R. G. Cox	06413
2	0899	Technical Library	09616
1	1109	R. J. Pryor	09216
1	1110	D. Womble	09214
1	9018	Central Technical Files	08945-1